(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 753 811 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
09.10.2002 Bulletin 2002/41

(51) Int Cl.[7]: **G06F 9/445**, G06F 9/46

(21) Application number: **96305139.6**

(22) Date of filing: **12.07.1996**

(54) Data processing method and device

Verfahren und Vorrichtung zur Datenverarbeitung

Appareil et procédé pour le traitement de données

(84) Designated Contracting States:
BE DE FR GB

(30) Priority: **14.07.1995 JP 17862595**

(43) Date of publication of application:
15.01.1997 Bulletin 1997/03

(60) Divisional application:
02077625.8

(73) Proprietor: **SONY CORPORATION**
Tokyo 141 (JP)

(72) Inventor: **Yokote, Yasuhiko**
Shinagawa-ku, Tokyo 141 (JP)

(74) Representative: **Pratt, Richard Wilson et al**
D. Young & Co,
21 New Fetter Lane
London EC4A 1DA (GB)

(56) References cited:
EP-A- 0 718 761
US-A- 5 195 130
US-A- 5 423 042

- **MOTOROLA TECHNICAL DEVELOPMENTS, vol. 18, March 1993, SCHAUMBURG, ILLINOIS US, pages 91-93, XP000349572 GREGORY L. CANNON ET AL: "DOWNLOADABLE PAGER FUNCTIONALITY"**
- **IBM TECHNICAL DISCLOSURE BULLETIN, vol. 36, no. 12, December 1993, NEW YORK US, pages 651-654, XP000419101 "Incremental Compilation for Interpretative Language Translator"**

EP 0 753 811 B1

Printed by Jouve, 75001 PARIS (FR)

Description

[0001] The invention relates to a data processing method, a server device, a client date processing device, a communications system, and computer software.

[0002] Recently, personal computers have become widespread, with these personal computers being used to access prescribed servers via a network so that prescribed information can be obtained.

[0003] Application programs are necessary for carrying out various processes with these kinds of personal computers. Each user purchases application programs which can operate with the operating system (hereinafter referred to as "OS") of their personal computer and installs and uses these application programs directly from a recording medium or via a communications line.

[0004] An idea for determining whether or not a client is appropriate while an application program is being installed via a communications line had been disclosed in Japanese Laid-open Patent No. Hei. 1-15253612. However, the object of this was to determine whether or not the user of the client computer is a regular user, with compatibility between the program and the computer not being determined in the related art at the time of installation of the program.

[0005] When OSs are different, the application program is also different, with each user then selecting and purchasing application programs in line with their own OS. The application providers (application designers) also then have to design a number (equal to the number of OSs) of application programs for carrying out what is essentially the same process, which involves a great deal of labour and at the same time increases costs.

[0006] A similar problem also occurs with application programs having the same OS. Namely, two application programs have to be designed separately even when both of these application programs are operating on the same OS when one of these application programs is different from the other application program. This makes the amount of effort and cost involved in providing a single application program high.

[0007] The prior art document EP-A-0718761 published on 26.06.1996 describes a system in which a client selects an object to view and a class loader determines whether a viewer application program appropriate for viewing the selected object is available at the client. If the class loader determines that the appropriate viewer application program is not available at the client, it locates an appropriate viewer on a server and downloads it to the client. If the downloaded viewer application program is in platform independent bytecode, the viewer is subjected to a verification procedure to verify its integrity prior to execution.

[0008] According to one aspect of the present invention there is provided a data processing method for a data processing system comprising: on a server side: storing an application program comprising a plurality of objects; storing an execution environment comprising a plurality of objects for specifying operations of said application program such that said execution environment is compatible with said application program; storing an application program interface operable to provide an interface between said execution environment and said application program; and on a client side: requesting a download of said application program from said server; wherein a checking means on said server is operable to execute a compatibility check to determine whether or not said client has said execution environment that is compatible with said requested application program and said requested application program is downloaded from said server in dependence upon a result of said check.

[0009] In a preferred embodiment, when said check by said server determines that said client does not have an execution environment with which said requested application program is compatible, said method comprises the steps of: downloading at least one execution environment object from said server to said client such that said compatible execution environment is constructed on said client; and downloading said application program from said server to said client.

[0010] According to an another aspect of the present invention, there is provided a server device for downloading an application program in response to a request from a client, said server device comprising: storage means storing an application program comprising one or more objects; storage means storing an execution environment comprising a plurality of objects for specifying operations of said application program wherein said execution environment is compatible with said application program; storage means storing an application program interface operable to provide an interface between said execution environment and said application program; checking means checking whether or not said client has an execution environment that is compatible with said requested application program; and downloading means for downloading said requested application program to said client in dependence upon a result of said check performed by said checking means.

[0011] According to a further aspect of the present invention, there is provided a client data processing device comprising: downloading means for downloading data from a server; storage means storing an application program comprising one or more objects; storage means storing an execution environment comprising a plurality of objects for specifying operations of said application program; storage means storing an application program interface operable provide an interface between said execution environment and said application program; notifying means for notifying said server as to whether an execution environment on said data processing device is compatible with an application program requested for download from said server; wherein said downloading means is operable to download said requested application program from said server in dependence upon said notification generated by said no-

tifying means.

[0012] These and other aspects of the invention are set out in the claims to which attention is invited.

[0013] The present invention thus provides a flexible software platform that allows application — independent inter-operability. Since the compatibility of an application program requested for download by the client can be checked with regard to the execution environment available on the client prior to downloading, the present invention affords the opportunity to construct an execution environment on the client with a core functionality and to achieve flexibility by downloading additional execution environment objects to support requested application programs as required. The required execution environment objects can be downloaded when it is determined that the requested application program is currently incompatible with the execution environment on the client. The construction of the execution environment as a collection of objects means that objects can be added to and removed from the execution environment as required by the application program and the application program interface that links the application program to the execution environment can be adapted accordingly. The advantages of the flexibility of being able to construct just the execution environment currently necessary at the client are explained in the description.

[0014] For a better understanding of the present invention, reference will now be made by way of example to the accompanying drawings in which:

FIG. 1 is a view of an example structure to which an illustrative data processing method of the present invention is applied;
FIG. 2 is a view showing a structure of an application program;
FIG. 3 is a view showing a structure of a concurrent object;
FIG. 4 is a view describing the downloading of objects from the server to clients of a plurality of vendors;
FIG. 5 is a view illustrating incremental downloading;
FIG. 6 is a view describing a meta-standard;
FIG. 7 is a view describing dynamic changing of an object;
FIG. 8 is a view describing dynamic expansion of an object;
FIG. 9 is a view describing the minimum functions of client;
FIG. 10 is a view describing the client environment structure suitable for an application and dynamic restructuring thereof;
FIG. 11 is a view describing a structure of a feature structure;
FIG. 12 is a view showing an example system structure to which the illustrative data processing device of the present invention is applied;
FIG. 13 is a view showing the logical structure of an MVM and an MK;
FIG. 14 is a view showing the logical structure of Context and Descriptor;
FIG. 15 is a view showing the overall structure of an MVM;
FIG. 16 is a view showing the data structure of Context and items surrounding Context;
FIG. 17 is a view showing the type of Variable table entry;
FIG. 18 is a view showing a primitive object list and interface name;
FIG. 19 is a view showing a primitive object interface;
FIG. 20 is a view showing a continuation of the primitive object interface in Fig. 19;
FIG. 21 is view showing a continuation of the primitive object interface in Fig. 20;
FIG. 22 is a view showing an I-code instruction set; and
FIG. 23 is a view showing an MK interface.

[0015] FIG. 1 shows an example of an illustrative system structure to which the data processing method of the present invention is applied. The system comprises a server 1 (data processing device), a client 2 (data processing device) and a network 3.

[0016] In this embodiment, the server 1 has two application programs 11-1 and 11-2. One of the application programs 11-1 has an execution environment 12-1 for defining the environment of execution of the application program 11-1 and an application program interface (hereinafter referred to as "API") 13-1 comprising an interface between the application program 11-1 and the execution environment 12-1.

[0017] The application program 11-1 comprises a plurality of objects 14-1 and the execution environment 12-1 comprises a plurality of objects 15-1.

[0018] The application program 11-2 also has an execution environment 12-2 defining the environment for this application program 11-2 and an API 13-2 functioning as an interface between the application program 11-2 and the execution environment 12-2.

[0019] Further, the application program 11-2 also comprises a plurality of objects 14-2 and the execution environment 12-2 comprises a plurality of objects 15-2.

[0020] The client 2 also has two application programs 21-1 and 21-2. The application program 21-1 has an execution environment 22-1 for defining the environment for this application program 21-1, and an API 23-1 as an interface between the application program 21-1 and the execution environment 22-1. The application program 21-1 comprises a plurality of objects 24-1 and the execution environment 22-1 comprises a plurality of objects 25-1.

[0021] The application program 21-2 also has an execution environment 22-2 defining the environment for the application program 21-2 and an API 23-2. The application program 21-2 comprises a plurality of objects

24-2 and the execution environment 22-2 comprises a plurality of objects 25-2.

[0022] The objects are all defined as concurrent objects which are processed concurrently with other objects. A plurality of APIs exist between the server 1 and the client 2 because one API set can be considered as one execution environment.

[0023] As shown in FIG. 1 and FIG. 2, the application program 11 comprises a plurality of objects 14 gathered together. Further, execution speed is increased by having the application program 11 processed concurrently by constructing the object 14 as a concurrent object. Because the object 14 is a replacement unit, objects with operational bugs or objects with performance problems etc. can be replaced with objects which do not have errors. Problems of the objects 14 can therefore be resolved without having to remodel the whole of the application program 11. Further, a new application program can be easily made by taking the object 14 as a part and combining the object 14 with an object taken as a part of an already existing application program.

[0024] This application program is a single service unit, and can be, for example, an application program for; only displaying image data from the server 1; retrieving image data using an VCR function; selecting a service using a menu; home shopping; a family account book linked with home shopping; tax calculations, etc.

[0025] Operativity for common features can then be obtained by sharing application programs between objects. For example, an editor for inputting data to a family account book application program and an editor for inputting data in a home shopping program can be shared.

[0026] Next, a description of a concurrent object will be given, with the configuration of the concurrent object being shown in FIG. 3. The object 14 which is a concurrent object comprises a method entry table 14A open to the outside public, a method code 14B, a memory area 14C for holding the object conditions and a simple thread 14D for executing the method. Only one execution context (also referred to as a "thread") exists at the concurrent object 14. The concurrent object 14 therefore receives one message and does not process messages arriving during the processing of this message until execution of this message which is currently being processed is completed.

[0027] Only providing one thread within the object has the following benefits.

(1) It is not necessary to be concerned about synchronization between a plurality of activities. Namely, it is no longer necessary to carry out such a process as to determine a sequence of access to the shared data using an instruction for synchronizing such as semaphore when shared data exists. In other words, the sequence of the message transmission to the object determines the sequence of access.

(2) As a result of this, program errors due to mistakes in the way of obtaining synchronization no longer occur and reusability of the object is improved.

(3) Synchronization errors can be prevented in a majority of cases by providing, for example, a device driver using this method.

(4) Device drivers can safely be replaced because synchronization errors due to replacing of the device drivers can also be prevented.

(5) Portions for the device driver other than portions for actually controlling the hardware can be provided independently from the OS. The program development period can therefore be made shorter because the amount of time taken up on program development for the device drivers can be shared.

(6) The description relating to execution control between objects can be removed from the description of the application program. For example, it is usually necessary for a thread execution control program to be incorporated within the application program with, for example, a method employing multi threads (when a plurality of threads are used). It is then necessary to rewrite the application program when the thread programming environment changes. However, if there is only one thread as in the present illustrative embodiment of the invention, it is not necessary to describe this portion at the application program. The application program therefore does not have to be rewritten even if the execution control method changes. The most appropriate execution control method for putting this concurrent object in this execution environment is provided by a system using dynamic expansion theory for objects.

(7) It is therefore not necessary to consider parallel processing when describing an application program. If concurrent objects are programmed, the system then carries out parallel processing with the hardware under the most appropriate execution control automatically after this because concurrent objects are the parallel processing units. In the related art, the generation of a number of processes or the generation of a number of threads had to be designated during programming. If this designation is made without taking into consideration the performance of the hardware, the application program then becomes dedicated to specific hardware.

[0028] With this system, the object is downloaded as necessary. An example of a system for downloading objects from the server 1 to the clients 2 of a plurality of vendors is shown in FIG. 4. Clients API 13 (13-1, 13-2) to be used by the respective vendors are realized by the execution environments 12 (12-1, 12-2).

[0029] When an object is downloaded to the client 2 (2-1, 2-2) it is determined whether or not an execution environment 22 (22-1, 22-2) the same as the execution environment 12 on the server 1 exists on the client 2. If

the same execution environment does exist, the object is downloaded. If not, downloading is carried out after an execution environment 22 the same as the execution environment 12 on the server 1 is constructed.

[0030] For example, in FIG. 4, when the object 14-1 of the application program 11-1 of the server 1 is downloaded as the object 24-1 of the application program 21-1 of the client 2-1, an object 25-1A corresponding to an object 15-1A of the execution environment 12-1 of the server 1 is necessary at the execution environment 22-1 of the client 2-1. Then, for example, the object 15-1B (checking means) of the execution environment 12-1 interrogates the object 25-1B (notifying means) of the execution environment 22-1 for the feature structure (to be described later). The object 15-1C (downloading means) of the execution environment 12-1 and the object 25-1C (downloading means) of the execution environment 22-1 then download the objects 15-1A and 15-1B of the execution environment 12-1 as the objects 25-1A and 25-1B of the execution environment 22-1 in accordance with this response.

[0031] In the related method, it is necessary for the object to be downloaded to be provided taking into account the client API. For example, when the client is a UNIX system, the same UNIX system may be being used on the server or it may be necessary to provide an object constructed of some kind of cross-developed environment. If the server and the client have to be equipped with the same execution environment, the client device usually has to be equipped with expensive calculation resources. For example, more memory has to be provided when compared with the case where a dedicated execution environment is provided and a high speed CPU (Central Processing Unit) has to be provided to guarantee sufficient execution speed, with this increasing the cost of the device.

[0032] With regards to this, according to the present illustrative embodiment of a system of the present invention, these problems are resolved by downloading an execution environment for the application program at the same time as downloading the application program. This is to say that by constructing just the execution environment 22 currently necessary at the client 2 at the client 2 unnecessary resources do not have to be prepared at the client 2. For example, if the client 2 does not require 3-D graphics then it is not necessary to prepare a library for this purpose.

[0033] Further, when a client is playing-back a movie image using VOD (Video On Demand), services (services which are not necessary when viewing movie images) for interacting with the user can be temporarily removed from the client with corresponding amount of calculation resources being able to be allotted to other work. These resources can then be used as a buffer for pre-fetch image data from the server 1. Services for interaction can then be downloaded from the server 1 when required.

[0034] The following have been considered as objects to be downloaded with the present illustrative system of the present invention.

(1) All application programs.
(2) Device driver groups (for example, MPEG drivers, ATM drivers, image control drivers etc.) for controlling hardware resources provided by the client.
(3) Object groups (for example, VCR command management, stream management, real-time scheduler, memory management, window management, downloading control, communication protocol management, execution management etc.) providing system services for application programs.

[0035] The most appropriate execution environment for the application program can then be constructed on the client by combining these application programs and object groups.

[0036] The server 1 can be a device for transmitting image data and application programs or a device for transmitting information to the client 2 via the network 3. The client 2 is a device for processing information transmitted from the server 1 and it is not usually necessary for the client 2 to be connected to the network 3. The most appropriate execution environment can therefore be prepared for every application program because the execution environment is provided every application program.

[0037] In the related art, it is necessary to make preliminary estimate about the characteristics of the application program when constructing the system. For example, when the application program has to deal with image data, a system service equipped with a VCR-like user interface for handling real-time scheduling and image data is necessary. Further, if the application program uses 3-D graphics, a library for this purpose has to be provided and the system tends to be expanded. Typical examples are UNIX (trademark) or Windows (trademark), in which an amount of memory necessary for the system is increased each time its version is improved. With the present illustrative system of the present invention, only a minimum of functions is provided for executing application programs and the problems of the systems of the related art are resolved.

[0038] By constructing an application program 11 as an assembly of a plurality of objects and by providing these objects as concurrent objects, concurrent execution is possible using each of the objects as a unit and the object can be downloaded at the same time as the application program is executed. At this time, by downloading the objects necessary in the execution of the application program incrementally as shown by the process in FIG. 5, the user can be made to feel as if the time it takes to download a single application had been much reduced.

[0039] For example, as shown in FIG. 5, when it is necessary to download objects 14-1-1 to 14-1-11 of the application program 11 of the server 1 as the objects

24-1-1 to 24-1-11 of application program 21 of the client 2, rather than downloading each object at random, objects 14-1-1 to 14-1-3 necessary first in the execution of the application program 21 are downloaded first as the objects 24-1-1 to 24-1-3.

[0040] If these three objects exist, the application program 21 can be activated and processing starts. While this processing is being executed, the remaining objects 14-1-4 to 14-1-11 are then sequentially downloaded in second to fourth steps as the objects 24-1-4 to 24-1-11 of the application program 21. These second to fourth downloads are also executed in the order of which object is necessary first.

[0041] When the objects 24-1-1 to 24-1-3 of the application program 21 are downloaded, the processing for the application program 21 has already been started. The user then can be made to feel as if all of the objects had been downloaded. The user therefore only has to be aware of the time necessary for downloading three objects, which is shorter than the time necessary for downloading eleven objects. In other words, the user is in practical terms not aware of the time for downloading eight of the objects, so that it is as if the time had been removed.

[0042] This can be also applied to the case of constructing the execution environment 22 described with reference to FIG. 4 (and also described later with reference to FIG. 10) on the client 2. In this case, the time for downloading all of the objects for the execution environment can be made to seem less to the user by first downloading only the objects necessary for executing the application program from the objects comprising the execution environment 22. This method can also be applied to the booting of the system.

[0043] Here, incremental downloading means downloading the application program or execution environment in units of objects or portions thereof comprising the application program or execution environment as necessary rather than downloading at one time. When application programs are downloaded in conventional personal computer communications, compressed application programs are downloaded at one time. The application program can therefore not be used until downloading has completely finished. Further, with, for example, system booting, up until now, start-up would take place from when all of the system had been read into memory. In the case of a UNIX disc-less workstation, the system does not start-up until all of the OS has been read-into memory from the server so that the system can therefore not be used until reading-out is completed. This is not the case, however, if incremental downloading is used.

[0044] This method is successfully applied to a server 1 and a Set Top Box (hereinafter referred to as "STB") taken as client 2 as follows. First, it is no longer necessary to wait impatiently until the system starts up as is the case for current personal computers because the STB can soon be used after turning on the power supply. As an STB is extremely prominent as a household electrical appliance, it is not desirable to have to make the user wait until the system starts up.

[0045] When the STB is turned on, the STB downloads and starts to execute the objects necessary first. The time the user has to wait will then be just the time taken to initially download these objects. The typical time for downloading an object is a few milliseconds to a few tens of milliseconds. This time can therefore be made to be insignificant to the user by using an appropriate user interface. The necessary objects are then downloaded concurrently with the activation of the system as the activation processes for the system proceed.

[0046] Limitations also occur when executing a plurality of application programs at the same time because plenty of calculation resources such as for the server 1 are not prepared at the client 2. For example, in the case where a VOD service is selected by a navigation application program for appreciation of a movie, resources (memory) occupied by the navigation application program can be used for the movie appreciation application once appreciation of the movie has started. These resources (the object for managing the memory) can then be downloaded again at the time when the navigation application again becomes necessary.

[0047] The "time when necessary" is the time when a message is sent with regards to the object. Namely, when the very first object downloaded sends a message to another object, this object that received the message is downloaded. The object sending the next message can then be downloaded beforehand by utilizing object interdependence and reference relationships. By carrying this out concurrently with execution of the application program, delays due to downloading at the time of message communications can then be made small. This then increases the efficiency of the incremental downloading.

[0048] The execution environments 12 and 22 are also assemblies of the objects 15 and 25 and the same operations as for the objects 14 and 24 of the application programs 11 and 21 are possible. A meta-object for controlling the downloading sequence specialized for the application programs 11 and 21 can therefore be prepared as one of the objects 15 and 25 (for example, the object 25-1C of FIG. 4 is taken as an meta-object). In this way, a download sequence of an object that the above object utilizes can be appointed as being suitable for a specific application and the time a user has to wait can be minimized using incremental downloading.

[0049] A function for downloading objects from the server 1 to the client 2, an accompanying function for checking compatibility of execution environments of the objects and a function for constructing execution environments are necessary to realize this system. These are taken as the basic functions for all of the devices (clients 2) provided in this system to have. In this specification, this function is referred to as a meta-standard. The API of the execution environment of the OS etc. can

be freely expanded with this meta-standard. The minimum standardization together with this expansion then will allow the API to comply with all types of applications from now on.

[0050] For example, with the system shown in FIG. 6, an OS having independent APIs is being operated at each of the servers 1-1 and 1-2 and each of the clients 2-1 and 2-2. Namely, an API 23-1 (API #1) for the application program 21-1 is constructed at the client 2-1 so as to correspond to the execution environment 22-1. Further, an API 23-2 (API #3) for the application program 21-2 for the execution environment 22-2 is constructed at the client 2-2. Because of this, APIs which correspond to these APIs is already prepared at the server that downloads the programs to the clients 2-1 and 2-2. In this embodiment, an API 13-1 for the application program 11-1 is constructed at the server 1-1 for the execution environment 12-1. This API 13-1 is taken to be the API (API #1) corresponding to API 23-1 (API #1) of the client 2-1.

[0051] Similarly, API 13-3 (API #3) corresponding to the application program 11-3 is formed at the execution environment 12-3. This API 13-3 is corresponds to API 23-2 (API #3) of the client 2-2.

[0052] Objects 15-1A to 15-1C, 15-3A to 15-3C, 25-1A to 25-1C and 25-2A to 25-2C are provided as objects corresponding to this meta-standard at the servers 1-1 and 1-2 and the clients 2-1 and 2-2. As a result, necessary objects can be properly downloaded from the servers 1-1 and 1-2 to the clients 2-1 and 2-2 according to the metal-standard protocal.

[0053] To standardize the OSs of the clients into a single OS has been the tendency in this field. However, as a result of defining meta-standard as above and providing APIs corresponding to each of the client APIs only on the server side, it becomes no longer necessary to decide a standard.

[0054] As a result of not stipulating one OS standard, the object realizing the system service including the application program can be constructed independently of the OS. Namely, software written for a certain execution environment can automatically be restructured for another separate execution environment using migration. This function has not been provided in conventional systems. For example, software written for UNIX will not operate on a Windows platform without being re-written. In order to realize this function with application level software, software capable of eliminating this software dependence is necessary. However, OS independence of objects for realizing system services including device drivers become possible using this method.

[0055] By downloading objects in this way, the object of the client 2 is changed as necessary as shown in FIG. 7, i.e. an already-existing object is removed from the client and a new object is downloaded from the server.

[0056] For example, with the embodiment of FIG. 7, the object 24A of the application program 21 of the client 2 has become unnecessary and is therefore removed. A new, necessary object 14A is then downloaded to the client 2 from the server 1.

[0057] In this way, the following becomes possible.

(1) The software can be updated. For example, when a bug is found in hardware control software, this object can be removed and can be replaced with a new object. Household electrical appliances equipped with computors are used by general consumers who are not computer specialists and software updating using an installer performed in some of computers is not appropriate. Updating of software in the way of the present illustrative embodiments of the present invention is therefore extremely effective.

(2) The product cycle can be made long. For example, models of television receivers are changed every year, but general consumers cannot buy a new television receiver every year. However, with the system of the present invention, the newest software functions can be provided to the user without having to change the television receiver and model changing of the receiver due to software function expansion is no longer necessary. This is also applied to STBs.

(3) Compliance with changes in requirements for user interfaces can be provided. For example, a user-friendly menu can be provided when the user first starts to use the device, with this then being changed to a more directly operated user interface when the user has got used to using the device. However, both procedures are not necessary to be brought over to the client side. Rather, the user interface matched the skill of the user at the time can be brought over to the client side. In this way, limited client resources can be effectively utilized.

[0058] Further, a downloaded object can expand the object of the client 2 as necessary as shown in FIG. 8. In the embodiment of FIG. 8, an execution environment 22-2 for receiving a new service is generated for the object 24-1B of the application program 21-1 at the client 2. The necessary objects 25-1A and 25-1B are then migrated (shifted) from the execution environment 22-1 so as to become the objects 25-2A and 25-2B at the execution environment 22-2. Other necessary objects 25-1C and 25-1D are also migrated to the execution environment 22-2.

[0059] To the application program 21-2, the object 24-1B of the application program 21-1 is migrated to become the object 24-2B.

[0060] In this way, for example, a new execution environment for the necessary expansion of real-time scheduling is generated at the client and necessary objects are shifted over to the new environment. The object can therefore be made capable of receiving a real-time scheduling service without necessitating any changes.

[0061] The following results are obtained as a result

of this.

(1) New functions can be dealt with without having to add any changes to application program objects. The application program life therefore becomes longer and reusability is improved. In the related method, changing of the execution environment meant the re-writing of the application program because dependent code for the execution environment was included in the application program.
(2) With regards to application programs included in the equipment, highly functional control software for the equipment including the user interfaces is the part wished to be re-used unless the model of the equipment is not largely changed or to have shortened development periods only by expanding software functions using already existing code. However, if the part includes code dependent on the execution environment, the work involved in re-utilization becomes complicated. With methods up until now, no countermeasures were taken for utilization of the part, but with the illustrative method of the present invention this work is made to be either automatic or minimal.

[0062] The illustrative system of the present invention can be applied specifically in the following way.

(1) Halting of the whole system can be prevented by downloading a memory protection function when the reliability of the application program is low.
(2) Services provided by service provider can be changed every STB vendor. For example, a service provider providing a movie of movie company A can change the characteristics of the image to be transmitted according to whether the image is received by the STB of company B or the STD of company C.
(3) The processing method of the system can be changed in accordance with the compression method of audio-visual data sent to an STB or the picture quality requested by the user. For example, the method taken when adjusting picture quality is different for MPEG data and JPEG data and it has therefore been necessary to change the processing method of the system. However, with the illustrative system of the present invention, the processing method is selected as necessary in accordance with the data format.

[0063] It is therefore not necessary for the client 2 to provides various functions beforehand because most of the functions for the system can be downloaded from the server 1 and the client 2 is therefore made to only provide a minimum of. functions needed. FIG. 9 shows the minimum of functions possessed by the client 2 of the system of the present invention. An execution environment 22-1 for the device driver, an execution environment 22-2 for the system object and an execution environment 22-3 for the execution environment are formed at the client 2 as minimum functions.

[0064] The device drivers necessary for being existed beforehand are an object 24-1A taken as an input driver for processing input, an object 24-1B taken as a timer driver for managing time and an object 24-1C taken as a screen driver for controlling displaying. The system objects are an object 24-2A taken as an input handler for managing input, an object 24-2B taken as boot protocol for managing activation, and an object 24-2C taken as a memory manager for managing memory. A more highly functional device drivers or system objects can be downloaded from the server 1.

[0065] FIG. 10 shows an embodiment of a dynamic structure of client environments suitable for application programs (video, game, shopping, etc.) sent out from the server. In FIG. 10, an execution environment 22-4 for shopping use is constructed on the client 2-2 so that a shopping application program 11-2 can be downloaded from the server 1. Further, when the client 2-2 changes the application program over from the shopping application program to a movie application program, an execution environment 22-3 for a movie application program 21-3 is constructed on the client 2-2 and the movie application program 11-1 is downloaded from the server 1.

[0066] For example, the following process can be considered.

(1) When a user selects a movie.
At this time, a navigation application 11-3 is downloaded from the server 1-2 to, for example, the execution environment 22-1 of the client 2-1 so that the desired movie can be selected and an object 15-3 for window management and for managing inputs from the user etc. is downloaded as an object 25-1 as the necessary environment for the navigation application 11-3.
(2) When a user is enjoying a movie.

[0067] At this time, an object 15-1 for video stream management, data pre-read buffer management and VCR functions etc. is downloaded from the execution environment 12-1 of the server 1-1 to, for example, the execution environment 22-3 of the client 2-2 as an object 25-3.

[0068] In the illustrative system of the present invention, the feature structure shown in FIG. 11 is introduced in order to present the execution environment of the client by downloading. This feature structure is checked when objects are downloaded from the server 1 to the client 2 and the necessary execution environments are constructed where the objects are downloaded.

[0069] First, the server 1 carries out a negotiation with the client 2 in a first phase (negotiation phase) for the possibility of an object migration between meta-object space of the server 1 and meta-object space of the client 2. In a second phase (transferring phase) the object is

then actually transferred.

[0070] Object migration is a meta-level process which transmits the calculation resources in-use using internal information of the object and objects related to this object if necessary. The internal information of the object is expressed by a meta-level object referred to as a descriptor. In reality, the descriptor holds the name of the meta-object managing the object. A typical descriptor holds the name of the meta-object managing memory segments for the object, the name of the meta-object performing execution control of two or more objects (scheduler) and the name of the meta-object managing the giving of names to objects, etc.

[0071] A check is made in the first phase (negotiation phase) for the possibility of shifting object. In some meta-object spaces, object migration is not desirable. For example, if an object (device driver) is shifted in a meta-object space for managing a device driver, this shifting becomes actually meaningless if a hardware device does not actually exist at the client 2. Further, a meta-object managing memory segments of the object using a virtual memory management structure cannot manage the memory segments even by carrying out shifting if a virtual memory management structure does not exist where the objects are shifted. The following method is therefore prepared for the migration protocol.

[0072] Feature* Descriptor::CanSpeak (Feature* pFeature).

[0073] In this method, a CanSpeak operation is executed for the descriptor within the meta-object space at the client 2.

[0074] At this time, a feature structure is passed from the server 1 as an argument. As a result, the client 2 returns a feature structure to the server 1 indicating that acceptance at the client 2 is possible. The server 1 then checks the feature structure returned from the client 2 so as to be able to know in which categories given below the compatibility of this meta-object space is included.

[0075] Compatibility can be classified into three categories, completely compatible, partially compatible and incompatible.

[0076] Complete compatibility means that the object can be completely executed even after being shifted. Partial compatibility means that some restrictions are put on execution of the object after being shifted. Incompatibility means that the object cannot be continued to be executed after being shifted.

[0077] Object migration is not carried out in the case of incompatibility. In the case of partial compatibility, whether or not migration is carried out is decided by the user. In reality, an exception is sent back to the user and a migration decision is made using the exception process routine. In the case of complete compatibility or partial compatibility when object migration can be carried out, migration is carried out in accordance with the contents of the feature structure returned previously.

[0078] Before the negotiation phase, an empty descriptor is generated at the meta-object space of the client 2 by the next operation.

Descriptor::Descriptor()

[0079] The previous CanSpeak method can be forwarded for this descriptor. At this time, necessary meta-object generation, reference generation and listing of necessary information are carried out on the basis of the feature structure information.

[0080] The process at the meta-level of the second phase is the shifting or transmitting of the meta-object corresponding to the transmitted descriptor. Here, shifting of the meta-object is the entering of this meta-object to the meta-object space of the client 2, i.e. being referenced from the descriptor. Further, transmitting of the meta-object means transmitting the data within the meta-object as a message for the meta-object (which is referenced from the descriptor) present at the meta-object space at the client 2.

[0081] The actual operation relating to the shifting and transmitting of the meta-object is executed in this second phase (shifting phase) utilizing the feature structure obtained by the negotiation phase.

[0082] The actual transferring and transmitting of the meta-object in the shifting phase is activated by the following method.

Descriptor & Descriptor::operator = (Descriptor & source)

[0083] The descriptor class is an abstract class which defines a common protocol relating to the shifting and transmitting of the meta-object. The contents of the descriptor referenced using the source are copied to this descriptor.

[0084] The actual procedure can be defined as a subclass of the descriptor class.

[0085] The next method is a method relating to transmitting of the meta-object. These protocols are mainly used by a migrator (a meta-object included within the meta-object space for carrying out object migration).

CanonicalContext& Context::asCanonical()

[0086] Converting a machine-dependent Context structure to a machine-independent form. This protocol is executed when the feature structure indicates that a direct conversion of Context is not possible.

- Context& Context::operator = (Context& source)
- Context& Context::operator = (CanonicalContext& source)

[0087] The Context currently referenced by this is initialized using the Context referenced by source.

- CanonicalSegment & Segment::asCanonical()

[0088] Converting a machine-dependent Segment structure to a non-machine-independent form. This protocol is executed when the feature structure indicates that a direct conversion of Context is not possible.

- Segment& Segment::operator = (Segment& source)
- Segment& Segment::operator = (CanonicalSegment& source)

[0089] The Segment currently referenced by this is initialized using the Context referenced by source and the necessary region of memory is copied.

[0090] FIG. 11 shows the configuration of the feature structure. As shown in FIG. 11, pointers of the object description and the environment description are described in the entry.

[0091] An object name, a pointer of the same structure as for the structure indicated by the pointer of the environment description and the resource requirement of this object are described at the structure indicated by the pointer of the object description.

[0092] Further, an environment name, resource information of client hardware, resource requirement of this environment and list of meta-objects constituting the execution environment are described at the structure indicated by the pointer of the environment description.

[0093] A specific example of the contents of a feature structure is as follows.

(1) Information relating to the object. real time processibility amount of required processor
(2) information relating to meta-object hardware meta-object

* processor type
* data format segment meta-object
* size
* expandability, compressibility
* management principle
* layout context meta-object
* register information
* number of temporary variables
* processor conditions
  mailer meta-objects
* message cue length
* number of available processor messages
* necessity of external mailer
* message transmission method
  external mailer meta-object
* message cue length
* number of available processor messages
* protocol
  scheduler meta-object
* object conditions
* scheduling principle
  management-dependent meta-object
* number of external names possessed

[0094] As described above, when each of the clients have different OSs, the OSs of each of the clients are determined from this feature structure and the server then downloads objects corresponding to these OSs.

[0095] FIG. 12 shows an example of an illustrative system structure to which the illustrative data processing system of the present invention is applied. A core 31 of this system comprises a Micro Virtual Machine (MVM) 31a (first execution means) and a Micro Kernel (MK) 31b (second execution means). The MVM 31a interprets and executes intermediate code (I-code) to be described later and can called-up a personality object (system object) using the functions of the MK31b as necessary.

[0096] Portions other than the core 31 shown in FIG. 12 can, for example, be downloaded from the server 1 using the aforementioned methods.

[0097] This system dynamically compiles I-code into native code (binary code, machine code) as necessary. Objects already compiled in native code can be executed but in this case, the personality object 33 (binary code generating means) is called-up using the functions of the MK 31b and provides a service to the applications 35.

[0098] The MVM 31a and the MK 31b comprising the core 31 shown in FIG. 12 are surrounded by device driver objects (Device drivers) 32 and Personality objects (Personality component objects) 33, which are in turn surrounded by a class system (Class Libraries) 34 prepared for carrying out application programming, which are in turn surrounded by application programs 35.

[0099] The layer of personality objects 33 allows this system to provide various OSs or virtual machines. For example, execution of BASIC programs is carried out by executing intermediate code obtained by compiling the BASIC program using personality objects for BASIC programming.

[0100] In the illustrative system of the present invention, programs are compiled to I-code (intermediate code) for managing the object method so as to obtain a high degree of portability. Although I-code is not designed on the presumption of being interpreted and executed (executing the program while interpreting the program), but is designed to be compiled as necessary into native code. However, the MVM interprets and executes the I-code when dynamic compiling of the I-code is difficult due to various limitations.

[0101] However, in most cases the I-code is compiled into native code and directly executed by a CPU constructed within the system. Deficiencies in real time processibility or loss of processing speed accompanying Virtual Machine execution is therefore negligible.

[0102] The I-code comprises two instruction sets (OP_M, OP_R) of sufficiently high abstractness, to be described later with reference to FIG. 22 so as to give a high degree of Inter-Operability. The semantics (the structure of meanings) of these instruction sets is strongly related to the interface of MK31b. This is to say

that the instruction sets are under the strong influence of the structure of the illustrative system of the present invention shown in FIG. 12. The illustrative system of the present invention therefore has a high degree of Portability and Inter-Operability in spite of assumption of the native code.

[0103] Next, the method for the MVM 31a and the MK 31b is described. First, the data structure assumed by the MVM 31a and the I-code format are stipulated.

[0104] FIG. 13 shows the logical structure of the MVM 31a and the MK 31b shown in FIG. 12. The logical structure of both the MVM 31a and the MK 31b is basically the same, with both comprising an active context 41, a message frame 42 and an execution engine 43. However, the MVM 31a supports execution using I-code and the MK 31b supports execution using native code.

[0105] In FIG. 13, active context 41 points to the Context (described later with reference to FIG. 14) currently being executed and the message frame 42 points to the main part of the message for the MVM 31a and the MK 31b comprising the core 31.

[0106] In the case of MK 31b, the place where the main part of the message exists depends on the implementation system with there being cases where this place is allotted to memory as a stack frame, allotted to a heap or allotted to registers of a number of CPUs. On the other hand, in the case of MVM 31a, the message frame points to an operand following the instruction code. Internal registers other than these registers may also be necessary depending on the implementation but these registers are independent of this method.

[0107] In FIG. 13, the execution engine 43 executes I-code and native code. Further, primitive objects 44 are included at the execution engine 43 of the MVM 31, with these primitive objects 44 being code for processing primitive objects to be described later with reference to FIG. 18.

[0108] FIG. 14 shows the logical structure of the Context and Descriptor. The context structure 51 showing one execution state of the program comprises the fields of object, class, method and meta-etc., with further fields of icode and minf being provided at the field method. This Context structure 51 holds the state of MVM 31a, corresponds to a CPU register, and is completely independent of the memory management and systems of communicating between programs etc.

[0109] The Context structure 51 is a Context primitive object, with each field being linked with prescribed information, as described later with reference to FIG. 16. Further, the Context structure 51 is deeply dependent on the implementation of the core 31 (MVM 31a and MK 31b) but only portions which are not dependent are shown in FIG. 14.

[0110] The important field at the Context structure 51 is the meta-field, with this field pointing to the Descriptor structure 52. The entries for the Descriptor structure 52 consist of three groups, #tag, context and select, with the API of the Personality object 33 being decided by this descriptor structure 52. Here, #tag expresses the API name and addresses of the API are expressed by context and selector.

[0111] FIG. 15 shows the whole structure of the MVM 31a. Basically, all of the necessary information can be referenced by following links from the Context structure 51. Namely, the Context structure 51 is linked to the object 61. This object 61 comprises a link (class pointer) to a class corresponding to the object 61 and an instance region (object dependent field). What kind of information is stored at the instance region or in what manner this information is laid-out depends on the implementation of the object.

[0112] The class 62 mainly holds the method. The class 62 comprises the name (class name), the portion depending on the conditions of implementation (class dependent field) and the link table (method table) to the I-method structure 63. The I-method structure 63 is the block primitive object and comprises a header (Header), an I-code and a variable table (variable table). Further, magic# is the management number (ID) of the MVM 31a. Basically, the I-code instruction operand refers to the target object via this variable table entry.

[0113] In FIG. 15, the gray portions (Context 51, I-method 63) are portions depending on the MVM 31a and are structures required when the MVM 31a translates and executes the I-code.

[0114] FIG. 16 shows structures linking-out from the Context structure 51. The object field of the Context structure 51 points to the object 61 and the class field points to the class-dependent field of class 62, with the icode of this method field pointing to the I-code of the I-method 63. The icode corresponds to the program counter and points to the program being executed. Further, vtable of the method field points to the variable table of the I-method 63.

[0115] The temporary field points to a region for temporarily saving data and the meta-field points to the descriptor 52 described above. Moreover, the class pointer of the object 61 points to class 62 and the method table of the class 62 points to I-method 63.

[0116] The variable table entry is a group consisting of type and value, with the value depending on the type.

[0117] The MVM 31a processes the type as shown in FIG. 17. When the type is T_PRIMITIVE, the value field refers to a primitive object. The kind of group of <P_CLASS, P_BODY> shown in FIG. 18, such as, for example, <P_INTEGER, immediate>, <P_STRING, address to heap> is stored at the value field. FIG. 18 shows the primitive object list together with the name of interface (listed in the column of P_CLASS). FIG. 19 to FIG. 21 show an example of the primitive object interface shown in FIG. 18 described using an Interface Definition Language (IDL) method. This IDL method is disclosed in "COBRA V2.0, July 1995, P3-1 to 3-36".

[0118] In FIG. 17, an ID for referring to the object is stored in the value field when the type is T_POINTER. This ID is the only value within the system with its posi-

tion being independent. An object position corresponding to the ID is then specified by a personality object 33.

[0119] FIG. 22 shows two instruction sets interpreted and executed by the MVM 31. In the structure of FIG. 12, the instruction set OP_M is instructions entering from outside to inside and the instruction set OP_R is instructions returning from inside to outside.

[0120] Namely, the instruction set OP_M executes the operation indicated by this first operand. This operation is processed by Personality object 33 or by a primitive object. Further, the compiler generates I-code in order to execute the applications 35 in a more efficient manner as a number of processes are processed by the primitive object. In this way, for example, the arithmetic operation for integer is processed by the leading primitive object of FIG. 18.

[0121] Next, the interface of MK 31b is specified. FIG. 23 shows the interface of the micro kernel (MK) 31b. The logical structures of the MVM 31a and the MK 31b are the same as that shown in FIG. 13. Here, MK 31b processes the instruction sets OP_M and OP_R shown in FIG. 22 as a system call.

[0122] The system constructed in the way described above can be applied to, for example, the client 2 (2-1, 2-2) shown in FIG. 4. The execution environment most suitable for executing a prescribed application can then be constructed on the client 2 by downloading an arbitrary portion of the portions other than the core 31 shown in FIG. 12 from the server 1.

[0123] As described above, the following can be considered as objects to be downloaded.

(1) All application programs.
(2) Device driver groups (for example, MPEG drivers, ATM drivers, image control drivers etc.) for controlling hardware resources provided by the client.
(3) Object groups (personality objects) providing system services for application programs (for example, VCR command management, stream management, real-time scheduler, memory management, window management, downloading control, communication protocol management, execution management etc.) providing system services for application programs.

[0124] The most appropriate execution environment for the application program can then be constructed on the client by combining these application programs and object groups.

[0125] For example, when it is wished to execute a BASIC program, the personality object 33 for BASIC and the BASIC program (application) 35 are downloaded from the server 1. This system is capable of providing a BASIC Virtual Machine using this personality object 33. The downloaded BASIC program is then temporarily compiled into intermediate code by the compiler and executed by the BASIC Virtual Machine. The BASIC program can alternatively be executed after having been compiled into native code in the way described above.

[0126] In the aforementioned embodiment, downloading of a prescribed object was carried out from the server to the client. However, illustrative embodiments of the present invention can also be applied to downloading of objects from a prescribed client to a prescribed server or to downloading of objects between servers or between clients.

[0127] Further, in the aforementioned embodiment, the I-code comprises two instructions, but the present invention is by no means limited in this respect.

[0128] According to illustrative embodiments of the data processing method and data processing device of the present invention, a check is made as to whether or not the client has the execution environment of the application program to be downloaded and the application program is then downloaded to the client on the basis of this check. The structure of the client can therefore be simplified and the cost can be reduced. This makes the providing of cheap application programs possible.

[0129] Moreover, with the illustrative data processing device of the present invention, notification is given to the server with regards to the execution environment for the application program to be downloaded, with the application program then being downloaded from the server on the basis of this notification. This makes low-cost devices with simplified structure to be feasible and also makes the providing of low-cost application programs possible.

[0130] With the illustrative data processing device and the illustrative data processing method of the present invention, an application program converted into intermediate code is interpreted and executed, or intermediate code is dynamically compiled and the generated binary code is executed. Intermediate code can therefore be gradually interpreted and executed when dynamic compiling is difficult. Still further, portable applications can be constructed by giving the intermediate code a simple structure.

Claims

1. A data processing method for a data processing system comprising :

   on a server (16) side:

   storing an application program (11-1) comprising a plurality of objects (14-1);
   storing an execution environment (12-1) comprising a plurality of objects (15-1) for specifying operations of said application program such that said execution environment is compatible with said application program;
   storing an application program interface (13-1 API) operable to provide an interface

between said execution environment and said application program; and
on a client (2) side:

requesting a download of said application program from said server (16);

wherein a checking means on said server is operable to execute a compatibility check to determine whether or not said client (2) has said execution environment that is compatible with said requested application program and said requested application program is downloaded from said server (16) in dependence upon a result of said check.

2. The method according to claim 1, wherein when said check by said server determines that said client does not have an execution environment with which said requested application program is compatible, said method comprises the steps of:

downloading at least one execution environment object from said server to said client such that said compatible execution environment is constructed on said client; and
downloading said application program from said server to said client.

3. The method according to claim 1 or claim 2, wherein said application program objects and said execution environment objects are concurrent objects and a single thread of execution control is provided within each object.

4. The method according to claim 1, 2 or 3, wherein said execution environment objects are downloaded incrementally from said server to said client in an order appropriate for the execution of said requested application program and wherein execution of said requested application program proceeds before all of the necessary execution environment objects have been downloaded.

5. The method according to claim 1, 2 or 3 wherein said requested application program objects are downloaded incrementally from said server to said client in a sequence appropriate for execution of said application program and one or more of said requested application program objects are downloaded to said client concurrently with execution of said requested application program by said client.

6. The method according to claim 1,2 or 3 wherein those of said application program objects and said execution environment objects that are required first for processing by said client during execution of said requested application program are downloaded first from said server to said client.

7. The method according to claim 4, 5 or 6, wherein a sequence for downloading said application program objects and/or said environment objects from said server to said client is specified by an execution environment object.

8. The method according to any one of the preceding claims, comprising the step of removing an existing application program object from said client and replacing said removed object by downloading a new object from said server.

9. The method according to any one of the preceding claims, comprising the step of constructing a second execution environment on said client for an application program object associated with said first mentioned execution environment and migrating at least one application program object from said first mentioned execution environment to said second execution environment.

10. The method according to any one of the preceding claims, wherein said execution environment for said client comprises:

a download function for downloading said requested application program object from said server to said client;
a checking function for checking whether an execution environment on said client is compatible with said requested object; and
a construction function for constructing an execution environment on said client that is compatible with said requested object;

wherein additional functions are downloaded from said server to said client as necessary.

11. The method according to any one of the preceding claims, wherein said client comprises:

an execution environment having device driver objects;
an execution environment having system objects; and
an execution environment having execution environment objects;

wherein said device driver objects, said system objects and said execution environment objects that are provided on said client are the functions necessary to create the corresponding environment and wherein more highly functional device driver objects or system objects are downloaded from said server to said client as necessary.

12. The method according to any one of the preceding claims, wherein said server passes a feature struc-

ture
to said client that points to a description of said requested object and further points to a description of said compatible execution environment and wherein said client uses said feature structure to perform said check.

13. A server device for downloading an application program in response to a request from a client, said server device comprising:

storage means storing an application program comprising one or more objects;
storage means storing an execution environment comprising a plurality of objects for specifying operations of said application program wherein said execution environment is compatible with said application program;
storage means storing an application program interface operable provide an interface between said execution environment and said application program;
checking means for checking whether or not said client has an execution environment that is compatible with said requested application program; and
downloading means for downloading said requested application program to said client in dependence upon a result of said check performed by said checking means.

14. The server device according to claim 13, wherein when said checking means determines that said client does not have an execution environment with which said requested application program is compatible, said downloading means downloads at least one execution environment object from said server to said client such that said compatible execution environment is constructed on said client; and said downloading means also downloads said application program from said server to said client.

15. The server device according to claim 13 or 14, wherein said application program objects and said execution environment objects are concurrent objects and a single thread of execution control is provided within each object.

16. The server device according to claim 13, 14 or 15, wherein said downloading means incrementally downloads said execution environment objects to said client in an order appropriate for the execution of said requested application program and wherein execution of said requested application program on said client proceeds before all of the necessary execution environment objects have been downloaded.

17. The server device according to any one of claims 13 to 16 wherein said downloading means incrementally downloads said requested application program objects to said client in a sequence appropriate for execution of said application program and said server downloads one or more of said requested application program objects to said client concurrently with execution of said requested application program by said client.

18. The server device according to any one of claims 13 to 17, wherein said downloading means first downloads to said client those of said application program objects and said execution environment objects that are required first for processing by said client during execution of said requested application program.

19. The server device according to any one of claims 13 to 17, wherein said downloading means downloads said application program objects and/or said environment objects from said server to said client in a sequence specified by an execution environment object.

20. The server device according to any one of claims 13 to 18, wherein said server device removes an existing application program object from said client and replaces said removed object by using said downloading means to download a new object from said server.

21. The server device according to any one of claims 13 to 19, wherein said server device is used to construct a second execution environment on said client for an application program object associated with said first mentioned execution environment and said client migrates at least one application program object from said first mentioned execution environment to said second execution environment.

22. The server device according to any one of claims 13 to 20, wherein said execution environment for said client comprises:

a download function for downloading said requested application program object from said server to said client;
a checking function for checking whether an execution environment on said client is compatible with said requested object; and
a construction function for constructing an execution environment on said client that is compatible with said requested object;
and wherein said downloading means downloads additional functions from said server to said client as necessary.

23. The server device according to any one of claims 13 to 21, wherein said client comprises:

an execution environment having device driver objects;
an execution environment having system objects; and
an execution environment having execution environment objects;

wherein said device driver objects, said system objects and said execution environment objects that are provided on said client are the functions necessary to create the corresponding environment and wherein said downloading means downloads more highly functional device driver objects or system objects from said server to said client as necessary.

24. The server device according to any one of claims 13 to 22, wherein said server device passes a feature structure to said client that points to a description of said requested object and further points to a description of said compatible execution environment and wherein said client uses said feature structure to perform said check.

25. A client data processing device comprising:

downloading means for downloading data from a server;
storage means storing an application program comprising one or more objects;
storage means storing an execution environment comprising a plurality of objects for specifying operations of said application program;
storage means storing an application program interface operable provide an interface between said execution environment and said application program; and
notifying means for notifying said server as to whether an execution environment on said data processing device is compatible with an application program requested for download from said server;

wherein said downloading means is operable to download said requested application program from said server in dependence upon said notification generated by said notifying means.

26. The client data processing device according to claim 25, wherein when said notifying means notifies said server that said client does not have an execution environment with which said requested application program is compatible;

said downloading means downloads at least one execution environment object from said server to said client such that said compatible execution environment is constructed on said client; and

said downloading means also downloads said application program from said server to said client.

27. The client data processing device according to claim 25 or 26, wherein said application program objects and said execution environment objects are concurrent objects and a single thread of execution control is provided within each object.

28. The client data processing device according to any one of claims 25 to 27, wherein said downloading means downloads said execution environment objects incrementally from said server to said client in an order appropriate for the execution of said requested application program and wherein execution of said requested application program on said client proceeds before all of the necessary execution environment objects have been downloaded.

29. The client data processing device according to any one of claims 25 to 28, wherein said downloading means incrementally downloads said requested application from said server to said client in a sequence appropriate for execution of said application program and one or more of said requested application program objects are downloaded to said client concurrently with execution of said requested application program by said client.

30. The client data processing device according to any one of claims 25 to 29
wherein said downloading means first downloads from said server to said client those of said application program objects and said execution environment objects that are required first for processing by said client during execution of said requested application program.

31. The client data processing device according to any one of claims 25 to 30, wherein said downloading means downloads said application program objects and/or said environment objects from said server to said client in a sequence specified by an execution environment object.

32. The client data processing device according to any one of claims 25 to 31, wherein an existing application program object is removed from said client and said downloading means replaces said removed object by downloading a new object from said server.

33. The client data processing device according to any one of claims 25 to 32, wherein a second execution environment is constructed on said client for an application program object associated with said first

mentioned execution environment and said client migrates at least one application program object from said first mentioned execution environment to said second execution environment.

34. The client data processing device according to any one of claims 25 to 33, wherein said execution environment for said client comprises:

a download function for downloading said requested application program object from said server to said client;
a checking function for checking whether an execution environment on said client is compatible with said requested object; and
a construction function for constructing an execution environment on said client that is compatible with said requested object;

wherein additional functions are downloaded from said server to said client as necessary.

35. The client data processing device according to any one of claims 25 to 34, wherein said client comprises:

an execution environment having device driver objects;
an execution environment having system objects; and
an execution environment having execution environment objects;

wherein said device driver objects, said system objects and said execution environment objects that are provided on said client are the functions necessary to create the corresponding environment and wherein more highly functional device driver objects or system objects are downloaded from said server to said client as necessary.

36. The client data processing device according to any one of claims 25 to 35, wherein said server passes a feature structure to said client that points to a description of said requested object and further points to a description of said compatible execution environment and wherein said client uses said feature structure to determine a result passed to said server by said notifying means.

37. A communications system comprising a server device according to any one of claims 13 to 24 and a client data processing device according to any one of claims 25 to 36.

38. Computer software having program code for carrying out a method according to any one of claims 1 to 13.

39. A data providing medium by which computer software according to claim 38 is provided.

40. A medium according to claim 39, the medium being a transmission medium.

41. A medium according to claim 39, the medium being a storage medium.

Patentansprüche

1. Datenverarbeitungsverfahren für ein Datenverarbeitungssystem umfassend auf der Seite eines Servers (16)
die Speicherung eines Anwendungsprogramm (11-1), welches eine Vielzahl von Objekten (14-1) aufweist,
die Speicherung einer Ausführungsumgebung (12-1), die eine Vielzahl von Objekten (15-1) zur Spezifizierung von Operationen des betreffenden Anwendungsprogramms aufweist, derart, dass die genannte Anwendungsumgebung mit dem genannten Anwendungsprogramm kompatibel ist,
die Speicherung einer Anwendungsprogramm-Schnittstelle (13-1 API), die derart betreibbar ist, dass eine Schnittstelle zwischen der genannten Anwendungsumgebung und dem genannten Anwendungsprogramm bereitgestellt ist,
und auf der Seite eines Clients (2)
die Anforderung eines Herunterladens des genannten Anwendungsprogramms von dem genannten Server (16),
wobei eine Prüfeinrichtung auf dem betreffenden Server derart betreibbar ist, dass eine Kompatibilitätsprüfung ausgeführt wird, um zu bestimmen, ob der genannten Client (2) die genannte Ausführungsumgebung aufweist oder nicht, die mit dem angeforderten Anwendungsprogramm kompatibel ist, und wobei das genannte angeforderte Anwendungsprogramm von dem Server (16) in Abhängigkeit von einem Ergebnis der betreffenden Prüfung heruntergeladen wird.

2. Verfahren nach Anspruch 1, wobei in dem Fall, dass die genannte Prüfung durch den Server bestimmt, dass der genannte Client nicht über eine Ausführungsumgebung verfügt, mit der das betreffende angeforderte Anwendungsprogramm kompatibel ist, das Verfahren die Schritte umfasst:

Herunterladen zumindest eines Ausführungsumgebungsobjekts von dem genannten Server zu dem genannten Client, derart, dass die genannte kompatible Ausführungsformungsumgebung auf dem betreffenden Client errichtet wird,
und Herunterladen des genannten Anwen-

dungsprogramms von dem genannten Server zu dem genannten Client.

3. Verfahren nach Anspruch 1 oder 2, wobei die genannten Anwendungsprogrammobjekte und die genannten Ausführungsumgebungsobjekte gleichzeitig vorhandene Objekte sind und ein einziger Ausführungssteuerungsfaden innerhalb jedes Objekts vorgesehen ist.

4. Verfahren nach Anspruch 1, 2 oder 3, wobei die genannten Ausführungsumgebungsobjekte von dem genannten Server zu dem genannten Client in einer für die Ausführung des angeforderten Anwendungsprogramms geeigneten Reihenfolge heruntergeladen werden und wobei die Ausführung des genannten angeforderten Anwendungsprogramms weitergeht, bevor sämtliche der erforderlichen Ausführungsumgebungsobjekte inkremental heruntergeladen worden sind.

5. Verfahren nach Anspruch 1, 2 oder 3, wobei die genannten angeforderten Anwendungsprogrammobjekte von dem genannten Server zu dem genannten Client in einer für die Ausführung des genannten Anwendungsprogramms geeigneten Sequenz inkremental heruntergeladen werden und wobei eines oder mehrere der angeforderten Anwendungsprogrammobjekte zu dem genannten Client gleichzeitig mit der Ausführung des betreffenden angeforderten Anwendungsprogramms durch den genannten Client heruntergeladen werden.

6. Verfahren nach Anspruch 1, 2 oder 3, wobei jene der genannten Anwendungsprogrammobjekte und der genannten Ausführungsumgebungsobjekte, die für die Verarbeitung durch den genannten Client während der Ausführung des betreffenden angeforderten Anwendungsprogramms zuerst erforderlich sind, von dem genannten Server zu dem genannten Client zuerst heruntergeladen werden.

7. Verfahren nach Anspruch 4, 5 oder 6, wobei eine Folge bzw. Sequenz zum Herunterladen der genannten Anwendungsprogrammobjekte und/oder der genannten Umgebungsobjekte von dem genannten Server zu dem genannten Client durch ein Ausführungsumgebungsobjekt spezifiziert wird.

8. Verfahren nach einem der vorhergehenden Ansprüche, umfassend den Schritt des Entfernens eines existierenden Anwendungsprogrammobjekts von dem genannten Client und des Ersetzens des entfernten Objekts durch Herunterladen eines neuen Objekts von dem genannten Server.

9. Verfahren nach einem der vorhergehenden Ansprüche, umfassend den Schritt des Errichtens einer zweiten Ausführungsumgebung auf dem genannten Client für ein Anwendungsprogrammobjekt, welches der zuerst erwähnten Ausführungsumgebung zugeordnet ist, und Migrieren zumindest eines Anwendungsprogramms von der betreffenden zuerst erwähnten Ausführungsumgebung in die genannte zweite Ausführungsumgebung.

10. Verfahren nach einem der vorhergehenden Ansprüche, wobei die genannte Ausführungsumgebung für den genannten Client umfasst:

    eine Herunterladefunktion zum Herunterladen des genannten angeforderten Anwendungsprogrammobjekts von dem genannten Server zu dem betreffenden Client,
    eine Prüffunktion zum Prüfen, ob eine Ausführungsumgebung auf dem genannten Client mit dem betreffenden angeforderten Objekt kompatibel ist,
    und eine Errichtungsfunktion zum Errichten einer Ausführungsumgebung auf dem genannten Client, die kompatibel ist mit dem betreffenden angeforderten Objekt,

    wobei zusätzliche Funktionen von dem genannten Server zu dem betreffenden Client soweit erforderlich heruntergeladen werden.

11. Verfahren nach einem der vorhergehenden Ansprüche, wobei der genannten Client umfasst:

    eine Ausführungsumgebung, die Vorrichtungstreiberobjekte aufweist,
    eine Ausführungsumgebung, die Systemobjekte aufweist,
    und eine Ausführungsumgebung, die Ausführungsumgebungsobjekte aufweist,

    wobei die genannten Vorrichtungstreiberobjekte, die genannten Systemobjekte und die genannten Ausführungsumgebungsobjekte, die auf dem betreffenden Client vorgesehen sind, die Funktionen sind, die zur Erzeugung der entsprechenden Umgebung notwendig sind, und wobei höhere funktionale Vorrichtungstreiberobjekte oder Systemobjekte von dem genannten Server zu dem genannten Client soweit erforderlich heruntergeladen werden.

12. Verfahren nach einem der vorhergehenden Ansprüche, wobei der genannte Server eine Merkmalsstruktur zu dem genannten Client weiterleitet, die auf eine Beschreibung des betreffenden angeforderten Objekts gerichtet ist und die ferner auf eine Beschreibung der genannten kompatiblen Ausführungsumgebung gerichtet ist, und wobei der genannten Client die betreffende Merkmalsstruktur zur Ausführung der genannten Prüfung nutzt.

13. Servervorrichtung zum Herunterladen eines Anwendungsprogramms auf eine Anforderung von einem Client hin, wobei die betreffende Servervorrichtung umfasst:

eine Speichereinrichtung, die ein Anwendungsprogramm speichert, welches aus einem oder mehreren Objekten besteht,
eine Speichereinrichtung, die eine Ausführungsumgebung speichert, welche eine Vielzahl von Objekten zur Spezifizierung von Operationen des betreffenden Anwendungsprogramms aufweist, wobei die betreffende Ausführungsumgebung mit dem genannten Anwendungsprogramm kompatibel ist,
eine Speichereinrichtung, die eine Anwendungsprogrammschnittstelle speichert, welche derart betreibbar ist, dass eine Schnittstelle zwischen der genannten Ausführungsumgebung und
dem genannten Anwendungsprogramm bereitgestellt ist,
eine Prüfeinrichtung zum Prüfen, ob der genannten Client über eine Ausführungsumgebung verfügt oder nicht, die mit dem betreffenden angeforderten Anwendungsprogramm kompatibel ist, und eine Herunterladeeinrichtung zum Herunterladen des genannten angeforderten Anwendungsprogramms zu dem genannten Client in Abhängigkeit von einem Ergebnis der durch die genannte Prüfeinrichtung durchgeführten Prüfung.

14. Servervorrichtung nach Anspruch 13, wobei dann, wenn die genannte Prüfeinrichtung bestimmt, dass der betreffende Client nicht über eine Ausführungsumgebung verfügt, mit der das betreffende angeforderte Anwendungsprogramm kompatibel ist, die genannte Herunterladeeinrichtung zumindest ein Ausführungsumgebungsobjekt von dem genannten Server zu dem genannten Client herunterlädt, derart, dass die betreffende kompatible Ausführungsumgebung auf dem genannten Client erstellt ist,
und wobei die genannte Herunterladeeinrichtung außerdem das genannte Anwendungsprogramm von dem betreffenden Server zu dem genannten Client herunterlädt.

15. Servervorrichtung nach Anspruch 13 oder 14, wobei die genannten Anwendungsprogrammobjekte und die genannten Ausführungsumgebungsobjekte gleichzeitig vorhandene Objekte sind und wobei ein einzelner Ausführungssteuerungsfaden in jedem Objekt vorgesehen ist.

16. Servervorrichtung nach Anspruch 13, 14 oder 15, wobei die genannte Herunterladeeinrichtung die genannten Ausführungsumgebungsobjekte zu dem betreffenden Client in einer für die Ausführung des angeforderten Anwendungsprogramms geeigneten Reihenfolge inkremental herunterlädt und wobei die Ausführung des betreffenden angeforderten Anwendungprogramms auf dem genannten Client weitergeht, bevor sämtliche der erforderlichen Ausführungsumgebungsobjekte heruntergeladen worden sind.

17. Servervorrichtung nach einem der Ansprüche 13 bis 16, wobei die genannte Herunterladeeinrichtung die angeforderten Anwendungsprogrammobjekte zu dem genannten Client in einer für die Ausführung des betreffenden Anwendungsprogramms geeigneten Sequenz inkremental herunterlädt und wobei der genannte Server eines oder mehrere der angeforderten Anwendungsprogrammobjekte zu dem betreffenden Client gleichzeitig mit der Ausführung des betreffenden angeforderten Anwendungsprogramms durch den genannten Client herunterlädt.

18. Servervorrichtung nach einem der Ansprüche 13 bis 17, wobei die genannte Herunterladeeinrichtung zuerst zu dem betreffenden Client jene der genannten Anwendungsprogrammobjekte und der genannten Ausführungsumgebungsobjekte herunterlädt, die für die Verarbeitung durch den genannten Client während der Ausführung des genannten angeforderten Anwendungsprogramms zuerst erforderlich sind.

19. Servervorrichtung nach einem der Ansprüche 13 bis 17, wobei die genannte Herunterladeeinrichtung die betreffenden Anwendungsprogrammobjekte und/oder die genannten Umgebungsobjekte von dem genannten Server zu dem betreffenden Client in einer durch ein Ausführungsumgebungsobjekt spezifizierten Sequenz herunterlädt.

20. Servervorrichtung nach einem der Ansprüche 13 bis 18, wobei die genannte Servervorrichtung ein existierendes Anwendungsprogrammobjekt von dem genannten Client entfernt und das entfernte Objekt durch Heranziehung der genannten Herunterladeeinrichtung zum Herunterladen eines neuen Objekts von dem genannten Server ersetzt.

21. Servervorrichtung nach einem der Ansprüche 13 bis 19, wobei die genannte Servervorrichtung dazu herangezogen wird, eine zweite Ausführungsumgebung auf dem genannten Client für ein Anwendungsprogrammobjekt zu errichten, welches der erstgenannten Ausführungsumgebung zugeordnet ist, und wobei der betreffende Client zumindest ein Anwendungsprogrammobjekt von der betreffenden ersterwähnten Ausführungsumgebung in die genannte zweite Ausführungsumgebung migriert.

22. Servervorrichtung nach einem der Ansprüche 13 bis 20, wobei die genannte Ausführungsumgebung für den genannten Client umfasst:

eine Herunterladefunktion zum Herunterladen des angeforderten Anwendungsprogrammobjekts von dem genannten Server zu dem genannten Client,
eine Prüffunktion zum Prüfen, ob eine Ausführungsumgebung auf dem betreffenden Client mit dem angeforderten Objekt kompatibel ist,
und eine Errichtungsfunktion zum Errichten einer Ausführungsumgebung auf dem betreffenden Client, die kompatibel ist mit dem betreffenden angeforderten Objekt,
und wobei die genannte Herunterladeeinrichtung zusätzliche Funktionen von dem genannten Server zu dem genannten Client soweit erforderlich herunterlädt.

23. Servervorrichtung nach einem der Ansprüche 13 bis 21, wobei der genannte Client umfasst:

eine Ausführungsumgebung, die Vorrichtungstreiberobjekte aufweist,
eine Ausführungsumgebung, die Systemobjekte aufweist,
und eine Ausführungsumgebung, die Ausführungsumgebungsobjekte aufweist,

wobei die betreffenden Vorrichtungstreiberobjekte, die genannten Systemobjekte und die genannten Ausführungsumgebungsobjekte, die auf dem betreffenden Client vorgesehen sind, die Funktionen sind, die zur Erzeugung der entsprechenden Umgebung erforderlich sind, und wobei die genannte Herunterladeeinrichtung höhere funktionale Vorrichtungstreiberobjekte oder Systemobjekte von dem genannten Server zu dem genannten Client soweit erforderlich herunterlädt.

24. Servervorrichtung nach einem der Ansprüche 13 bis 22, wobei die genannte Servervorrichtung eine Merkmalsstruktur zu dem betreffenden Client weiterleitet, die auf eine Beschreibung des genannten angeforderten Objekts gerichtet ist und die ferner auf eine Beschreibung der betreffenden kompatiblen Ausführungsumgebung gerichtet ist, und wobei der genannte Client die betreffende Merkmalsstruktur zur Ausführung der genannten Prüfung heranzieht.

25. Client-Datenverarbeitungsvorrichtung umfassend:

eine Herunterladeeinrichtung zum Herunterladen von Daten von einem Server,
eine Speichereinrichtung, die ein Anwendungsprogramm speichert, welches ein oder mehrere Objekte aufweist,
eine Speichereinrichtung, die eine Ausführungsumgebung speichert, welche eine Vielzahl von Objekten zur Spezifizierung von Operationen des betreffenden Anwendungsprogramms aufweist,
eine Speichereinrichtung, die eine Anwendungsprogrammschnittstelle speichert, welche derart betreibbar ist, dass eine Schnittstelle zwischen der genannten Ausführungsumgebung und dem genannten Anwendungsprogramm bereitgestellt ist,
und eine Meldeeinrichtung, die dem betreffenden Server meldet, ob eine Ausführungsumgebung auf der betreffenden Datenverarbeitungsvorrichtung mit einem Anwendungsprogramm kompatibel ist, welches zum Herunterladen von dem betreffenden Server angefordert ist,

wobei die genannte Herunterladeeinrichtung derart betreibbar ist, dass das betreffende angeforderte Anwendungsprogramm von dem genannten Server in Abhängigkeit von der durch die genannte Meldeeinrichtung erzeugten Meldung heruntergeladen wird.

26. Client-Datenverarbeitungsvorrichtung nach Anspruch 25,
wobei dann, wenn die genannte Meldeeinrichtung dem Server meldet, dass der betreffende Client nicht über eine Ausführungsumgebung verfügt, mit der das angeforderte Anwendungsprogramm kompatibel ist, die genannte Herunterladeeinrichtung zumindest ein Ausführungsumgebungsobjekt von dem genannten Server zu dem betreffenden Client herunterlädt, derart, dass die betreffende kompatible Ausführungsumgebung auf dem betreffenden Client errichtet wird,
und die genannte Herunterladeeinrichtung außerdem das genannte Anwendungsprogramm von dem genannten Server zu dem genannten Client herunterlädt.

27. Client-Datenverarbeitungsvorrichtung nach Anspruch 25 oder 26, wobei die genannten Anwendungsprogrammobjekte und die genannten Ausführungsumgebungsobjekte gleichzeitig vorhandene Objekte sind und wobei ein einzelner Ausführungssteuerungsfaden in jedem Objekt vorgesehen ist.

28. Client-Datenverarbeitungsvorrichtung nach einem der Ansprüche 25 bis 27, wobei die genannte Herunterladeeinrichtung die Ausführungsumgebungsobjekte von dem genannten Server zu dem betreffenden Client in einer für die Ausführung des betreffenden angeforderten Anwendungsprogramms ge-

eigneten Reihenfolge inkremental herunterlädt und wobei die Ausführung des betreffenden angeforderten Anwendungsprogramms auf dem genannten Client weitergeht, bevor sämtliche der notwendigen Ausführungsumgebungsobjekte heruntergeladen worden sind.

29. Client-Datenverarbeitungsvorrichtung nach einem der Ansprüche 25 bis 28, wobei die genannte Herunterladeeinrichtung die angeforderte Anwendung von dem genannten Server zu dem betreffenden Client in einer für die Ausführung des genannten Anwendungsprogramms geeigneten Sequenz inkremental herunterlädt und wobei ein oder mehrere der betreffenden angeforderten Anwendungsprogrammobjekte zu dem genannten Client gleichzeitig mit der Ausführung des betreffenden angeforderten Anwendungsprogramms durch den genannten Client heruntergeladen werden.

30. Client-Datenverarbeitungsvorrichtung nach einem der Ansprüche 25 bis 29, wobei die genannte Herunterladeeinrichtung zuerst von dem genannten Server zu dem betreffenden Client jene der genannten Anwendungsprogrammobjekte und der genannten Ausführungsumgebungsobjekte herunterlädt, die für die Verarbeitung durch den betreffenden Client während der Ausführung des genannten angeforderten Anwendungsprogramms zuerst erforderlich sind.

31. Client-Datenverarbeitungsvorrichtung nach einem der Ansprüche 25 bis 30, wobei die genannte Herunterladeeinrichtung die genannten Anwendungsprogrammobjekte und/oder die genannten Umgebungsobjekte von dem genannten Server zu dem betreffenden Client in einer durch ein Ausführungsumgebungsobjekt spezifizierten Sequenz herunterlädt.

32. Client-Datenverarbeitungsvorrichtung nach einem der Ansprüche 25 bis 31, wobei ein existierendes Anwendungsprogrammobjekt von dem betreffenden Client entfernt wird und wobei die genannte Herunterladeeinrichtung das betreffende entfernte Objekt durch Herunterladen eines neuen Objekts von dem genannten Server ersetzt.

33. Client-Datenverarbeitungsvorrichtung nach einem der Ansprüche 25 bis 32, wobei eine zweite Ausführungsumgebung auf dem betreffenden Client für ein Anwendungsprogrammobjekt errichtet wird, welches der ersterwähnten Ausführungsumgebung zugeordnet ist, und wobei der betreffende Client zumindest ein Anwendungsprogrammobjekt von der betreffenden ersterwähnten Ausführungsumgebung in die genannte zweite Ausführungsumgebung migriert.

34. Client-Datenverarbeitungsvorrichtung nach einem der Ansprüche 25 bis 33, wobei die genannte Ausführungsumgebung für den genannten Client umfasst:

eine Herunterladefunktion zum Herunterladen des genannten angeforderten Anwendungsprogrammobjekts von dem genannten Server zu dem betreffenden Client,
eine Prüffunktion zum Prüfen, ob eine Ausführungsumgebung auf dem genannten Client mit dem betreffenden angeforderten Objekt kompatibel ist,
und eine Errichtungsfunktion zum Errichten einer Ausführungsumgebung auf dem genannten Client, die kompatibel ist mit dem betreffenden angeforderten Objekt,

wobei zusätzliche Funktionen von dem genannten Server zu dem betreffenden Client soweit erforderlich heruntergeladen werden.

35. Client-Datenverarbeitungsvorrichtung nach einem der Ansprüche 25 bis 34, wobei der genannte Client umfasst:

eine Ausführungsumgebung, die Vorrichtungstreiberobjekte aufweist,
eine Ausführungsumgebung, die Systemobjekte aufweist,
und eine Ausführungsumgebung, die Ausführungsumgebungsobjekte aufweist,

wobei die genannten Vorrichtungstreiberobjekte, die genannten Systemobjekte und die genannten Ausführungsumgebungsobjekte, die auf dem betreffenden Client vorgesehen sind, die Funktionen sind, die zur Erzeugung der entsprechenden Umgebung notwendig sind, und wobei höhere funktionale Vorrichtungstreiberobjekte oder Systemobjekte von dem genannten Server zu dem genannten Client soweit erforderlich heruntergeladen werden.

36. Client-Datenverarbeitungsvorrichtung nach einem der Ansprüche 25 bis 35, wobei der genannte Server eine Merkmalsstruktur zu dem betreffenden Client weiterleitet, die auf eine Beschreibung des betreffenden angeforderten Objekts gerichtet ist und die ferner auf eine Beschreibung der betreffenden kompatiblen Ausführungsumgebung gerichtet ist und wobei der genannte Client die betreffende Merkmalsstruktur zur Bestimmung eines Ergebnisses heranzieht, das durch die genannte Meldeeinrichtung zu dem genannten Server hin geleitet ist.

37. Kommunikationssystem, umfassend eine Servervorrichtung nach einem der Ansprüche 13 bis 24 und eine Client-Datenverarbeitungsvorrichtung

nach einem der Ansprüche 25 bis 36.

38. Computer-Software mit einem Programmcode zur Ausführung eines Verfahrens gemäß einem der Ansprüche 1 bis 13.

39. Datenbereitstellungsmedium, durch das Computer-Software nach Anspruch 38 bereitgestellt wird.

40. Medium nach Anspruch 39, wobei das Medium ein Übertragungsmedium ist.

41. Medium nach Anspruch 39, wobei das Medium ein Speichermedium ist.

Revendications

1. Procédé de traitement de données pour un système de traitement de données comprenant :

du côté du serveur (16) :

le stockage d'un programme d'application (11-1) comprenant une pluralité d'objets (14-1) ;
le stockage d'un environnement d'exécution (12-1) comprenant une pluralité d'objets (15-1) pour spécifier des opérations dudit programme d'application pour que ledit environnement d'exécution soit compatible avec ledit programme d'application ;
stockage d'une interface de programme d'application (13-1 API) utilisable pour fournir une interface entre ledit environnement d'exécution et ledit programme d'application ; et
du côté du client (2) :

demande d'un téléchargement dudit programme d'application à partir dudit serveur (16) ;

où un moyen de contrôle sur ledit serveur est utilisable pour exécuter un contrôle de compatibilité pour déterminer si oui ou non ledit client (2) possède ledit environnement d'exécution qui est compatible avec ledit programme d'application demandé et ledit programme d'application demandé est téléchargé à partir dudit serveur (16) en fonction du résultat dudit contrôle.

2. Procédé selon la revendication 1, dans lequel ledit contrôle par ledit serveur détermine que ledit client n'a pas un environnement d'exécution avec lequel ledit programme d'application demandé est compatible, ledit procédé comprenant les étapes de :

téléchargement d'au moins un objet d'environnement d'exécution à partir dudit serveur au niveau dudit client pour que ledit environnement d'exécution compatible soit construit au niveau dudit client ; et
téléchargement dudit programme d'application à partir dudit serveur au niveau dudit client.

3. Procédé selon la revendication 1 ou 2, dans lequel lesdits objets de programme d'application et
lesdits objets d'environnement d'exécution sont des objets concurrents et un chemin unique de contrôle d'exécution est fourni dans chaque objet.

4. Procédé selon la revendication 1, 2, ou 3, dans lequel lesdits objets d'environnement d'exécution sont téléchargés de façon incrémentale à partir dudit serveur au niveau dudit client dans un ordre approprié pour l'exécution dudit programme d'application demandé et où l'exécution dudit programme d'application demandé avance avant que tous les objets d'environnement d'exécution nécessaires aient été téléchargés.

5. Procédé selon la revendication 1, 2 ou 3 dans lequel lesdits objets de programme d'application demandé sont téléchargés de façon incrémentale à partir dudit serveur au niveau dudit client dans une séquence appropriée pour l'exécution dudit programme d'application et un ou plusieurs desdits objets de programme d'application demandé sont téléchargés au niveau dudit client concurremment avec l'exécution dudit programme d'application demandé par ledit client.

6. Procédé selon la revendication 1, 2 ou 3 dans lequel ceux desdits objets de programme d'application et desdits objets d'environnement d'exécution qui sont exigés d'abord pour le traitement par ledit client pendant l'exécution dudit programme d'application demandé sont téléchargés d'abord à partir dudit serveur au niveau dudit client.

7. Procédé selon la revendication 4, 5 ou 6 dans lequel une séquence pour télécharger lesdits objets de programme d'application et/ou lesdits objets d'environnement à partir dudit serveur au niveau dudit client est spécifié par un objet d'environnement d'exécution.

8. Procédé selon une quelconque des revendications précédentes, comprenant l'étape d'enlèvement d'un objet de programme d'application existant à partir dudit client et remplacement dudit objet enlevé par le téléchargement d'un nouvel objet à partir dudit serveur.

9. Procédé selon l'une quelconque des revendications

précédentes, comprenant l'étape de construction dudit second environnement d'exécution au niveau dudit client pour un objet de programme d'application associé audit premier environnement d'exécution mentionné et migration au moins un objet de programme d'application à partir dudit premier environnement d'exécution mentionné vers ledit second environnement d'exécution.

10. Procédé selon l'une quelconque des revendications précédentes, dans lequel ledit environnement d'exécution pour ledit client comprend :

une fonction de téléchargement pour télécharger ledit objet de programme d'application demandé à partir dudit serveur au niveau dudit client ;
une fonction de contrôle pour contrôler si un environnement d'exécution au niveau dudit client est compatible avec ledit objet demandé ; et
une fonction de construction pour construire un environnement d'exécution au niveau dudit client qui est compatible avec ledit objet demandé ;

dans lequel des fonctions supplémentaires sont téléchargées à partir dudit serveur au niveau dudit client si nécessaire.

11. Procédé selon l'une quelconque des revendications précédentes, dans lequel ledit client comprend :

un environnement d'exécution ayant des objets de pilote de dispositif ;
un environnement d'exécution ayant des objets de système ; et
un environnement d'exécution ayant des objets d'environnement d'exécution ;

où lesdits objets de commande de dispositif, lesdits objets de système et lesdits objets d'environnement d'exécution qui sont fournis au niveau dudit client sont les fonctions nécessaires pour créer l'environnement correspondant et où des objets de pilote de dispositif ou des objets de système plus hautement fonctionnels sont téléchargés à partir dudit serveur au niveau dudit client si nécessaire.

12. Procédé selon l'une quelconque des revendications précédentes, dans lequel ledit serveur passe une structure de caractéristique au niveau dudit client qui pointe vers une description dudit objet demandé et pointe en outre vers une description dudit environnement d'exécution compatible et où ledit client utilise ladite structure de caractéristique pour réaliser ledit contrôle.

13. Dispositif de serveur pour télécharger un programme d'application en réponse à une demande d'un client, ledit dispositif de serveur comprenant :

un moyen de stockage stockant un programme d'application comprenant un ou plusieurs objets ;
un moyen de stockage stockant un environnement d'exécution comprenant une pluralité d'objets pour spécifier des opérations dudit programme d'application

où ledit environnement d'exécution est compatible avec ledit programme d'application ;
un moyen de stockage stockant une interface de programme d'application utilisable fournit une interface entre ledit environnement d'exécution et ledit programme d'application ;
un moyen de contrôle pour contrôler si oui ou non ledit client a un environnement d'exécution qui est compatible avec ledit programme d'application demandé ; et
un moyen de téléchargement pour télécharger ledit programme d'application demandé au niveau dudit client en fonction du résultat dudit contrôle réalisé par ledit moyen de contrôle.

14. Dispositif de serveur selon la revendication 13, dans lequel lorsque ledit moyen de contrôle détermine que ledit client n'a pas un environnement d'exécution avec lequel ledit programme d'application demandé est compatible, ledit moyen de téléchargement télécharge au moins un objet d'environnement d'exécution à partir dudit serveur au niveau dudit client pour que ledit environnement d'exécution compatible soit construit au niveau dudit client ; et ledit moyen de téléchargement télécharge aussi ledit programme d'application à partir dudit serveur au niveau dudit client.

15. Dispositif de serveur selon la revendication 13 ou 14, dans lequel lesdits objets de programme d'application et lesdits objets d'environnement d'exécution sont des objets concurrents et un chemin unique de la commande d'exécution est fourni dans chaque objet.

16. Dispositif de serveur selon la revendication 13, 14 ou 15, dans lequel ledit moyen de téléchargement télécharge incrémentalement lesdits objets d'environnement d'exécution au niveau dudit client dans un ordre approprié pour l'exécution par ledit programme d'application demandé et où l'exécution dudit programme d'application demandé au niveau dudit client avant que tous les objets de l'environnement d'exécution nécessaire aient été téléchargés.

17. Dispositif serveur selon l'une quelconque des re-

vendications 13 à 16 dans lequel le moyen de téléchargement télécharge incrémentalement lesdits objets de programme d'application demandé au niveau dudit client dans une séquence appropriée pour l'exécution dudit programme d'application et ledit serveur télécharge un ou plusieurs desdits objets de programme d'application demandé au niveau dudit client concurremment avec l'exécution dudit programme d'application demandé par ledit client.

18. Dispositif de serveur selon l'une quelconque des revendications 13 à 17, dans lequel ledit premier moyen de téléchargement télécharge au niveau dudit client ceux desdits objets de programme d'application et desdits objets d'environnement d'exécution qui sont demandés d'abord pour le traitement par ledit client pendant l'exécution dudit programme d'application demandé.

19. Dispositif de serveur selon l'une quelconque des revendications 13 à 17, dans lequel ledit moyen de téléchargement télécharge, lesdits objets de programme d'application et/ou lesdits objets d'environnement à partir dudit serveur au niveau dudit client dans une séquence spécifiée par un objet d'environnement d'exécution.

20. Dispositif de serveur selon l'une quelconque des revendications 13 à 18, dans lequel ledit dispositif de serveur enlève un objet de programme d'application existant à partir dudit client et remplace ledit objet enlevé en utilisant ledit moyen de téléchargement pour télécharger un nouvel objet à partir dudit serveur.

21. Dispositif de serveur selon l'une quelconque des revendications 13 à 19, dans lequel ledit dispositif de serveur est utilisé pour construire un second environnement d'exécution au niveau dudit client pour un objet de programme d'application associé audit premier environnement d'exécution mentionné et ledit client migre au moins un objet de programme d'application à partir dudit premier environnement d'exécution mentionné vers ledit second environnement d'exécution.

22. Dispositif de serveur selon l'une quelconque des revendications 13 à 20, dans lequel ledit environnement d'exécution pour ledit client comprend :

une fonction de téléchargement pour télécharger ledit objet de programme d'application demandé à partir dudit serveur au niveau dudit client ;
une fonction de contrôle pour contrôler si un environnement d'exécution au niveau dudit client est compatible avec ledit objet demandé ; et
une fonction de construction pour construire un environnement d'exécution au niveau dudit client qui est compatible avec ledit objet demandé ;
et dans lequel ledit moyen de téléchargement télécharge des fonctions supplémentaires à partir dudit serveur au niveau dudit client si nécessaire.

23. Dispositif de serveur selon l'une quelconque des revendications 13 à 21, dans lequel ledit client comprend :

un environnement d'exécution ayant des objets de pilote de dispositif ;
un environnement d'exécution ayant des objets de système ; et
un environnement d'exécution ayant des objets d'environnement d'exécution ;

dans lequel lesdits objets de commande de dispositif, lesdits objets de système et lesdits objets d'environnement d'exécution qui sont fournis au niveau dudit client sont les fonctions nécessaires pour créer l'environnement correspondant et où ledit moyen de téléchargement télécharge des objets de pilote de dispositif ou des objets de système plus hautement fonctionnels à partir dudit serveur au niveau dudit client si nécessaire.

24. Dispositif de serveur selon l'une quelconque des revendications 13 à 22, dans lequel ledit dispositif de serveur passe une structure de caractéristique au niveau dudit client qui pointe vers une description dudit objet demandé et pointe en outre vers une description dudit environnement d'exécution compatible et où ledit client utilise ladite structure de caractéristique pour réaliser ledit contrôle.

25. Dispositif de traitement de données de client comprenant :

un moyen de téléchargement pour télécharger des données à partir d'un serveur ;
un moyen de stockage stockant un programme d'application comprenant un ou plusieurs objets ;
un moyen de stockage stockant un environnement d'exécution comprenant une pluralité d'objets pour spécifier des opérations dudit programme d'application ;
un moyen de stockage stockant une interface de programme d'application utilisable pour fournir une interface entre ledit environnement d'exécution et ledit programme d'application ; et
un moyen de notification pour notifier audit serveur si un environnement d'exécution sur ledit

dispositif de traitement de données est compatible avec un programme d'application demandé pour le téléchargement à partir dudit serveur ;

dans lequel ledit moyen de téléchargement est utilisable pour télécharger ledit programme d'application demandé à partir dudit serveur en dépendance de ladite notification générée par ledit moyen de notification.

26. Dispositif de traitement de données de client selon la revendication 25, dans lequel lorsque ledit moyen de notification notifie audit serveur que ledit client n'a pas un environnement d'exécution avec lequel ledit programme d'application demandé est compatible ;

ledit moyen de téléchargement télécharge au moins un objet d'environnement d'exécution à partir dudit serveur au niveau dudit client pour que ledit environnement d'exécution compatible soit construit au niveau dudit client ; et

ledit moyen de téléchargement télécharge aussi ledit programme d'application à partir dudit serveur au niveau dudit client.

27. Dispositif de traitement de données de client selon la revendication 25 ou 26, dans lequel lesdits objets de programme d'application et lesdits objets d'environnement d'exécution sont des objets concurrents et un chemin unique de commande d'exécution est fourni dans chaque objet.

28. Dispositif de traitement de données de client selon l'une quelconque des revendications 25 à 27, dans lequel ledit moyen de téléchargement télécharge lesdits objets d'environnement d'exécution de façon incrémentale à partir dudit serveur au niveau dudit client dans un ordre approprié pour l'exécution dudit programme d'application demandé et où l'exécution dudit programme d'application demandé au niveau dudit client avance avant que tous les objets d'environnement d'exécution nécessaires aient été téléchargés.

29. Dispositif de traitement de données de client selon l'une quelconque des revendications 25 à 28, dans lequel ledit moyen de téléchargement télécharge incrémentalement ladite application demandée à partir dudit serveur au niveau dudit client dans une séquence appropriée pour l'exécution dudit programme d'application et un ou plusieurs desdits objets de programme d'application demandé sont téléchargés au niveau dudit client concurremment avec l'exécution dudit programme d'application demandé par ledit client.

30. Dispositif de traitement de données de client selon l'une quelconque des revendications 25 à 29, dans lequel ledit moyen de téléchargement télécharge d'abord à partir dudit serveur au niveau dudit client ceux desdits objets de programme d'application et desdits objets d'environnement d'exécution qui sont demandés d'abord pour le traitement par ledit client pendant l'exécution dudit programme d'application demandé.

31. Dispositif de traitement de données de client selon l'une quelconque des revendications 25 à 30, dans lequel ledit moyen de téléchargement télécharge lesdits objets de programme d'application et/ou lesdits objets d'environnement à partir dudit serveur au niveau dudit client dans une séquence spécifiée par un objet d'environnement d'exécution.

32. Dispositif de traitement de données de client selon l'une quelconque des revendications 25 à 31, dans lequel un objet de programme d'application existant est enlevé dudit client et ledit moyen de téléchargement remplace ledit objet enlevé en téléchargeant un nouvel objet dudit serveur.

33. Dispositif de traitement de données de client selon l'une quelconque des revendications 25 à 32, dans lequel un second environnement d'exécution est construit au niveau dudit client pour un objet de programme d'application associé audit premier environnement d'exécution mentionné et ledit client migre au moins un objet de programme d'application à partir dudit premier environnement d'exécution mentionné au niveau dudit second environnement d'exécution.

34. Dispositif de traitement de données de client selon l'une quelconque des revendications 25 à 33, dans lequel ledit environnement d'exécution pour ledit client comprend :

une fonction de téléchargement pour télécharger ledit objet de programme d'application demandé à partir dudit serveur au niveau dudit client ;
une fonction de contrôle pour contrôler si un environnement d'exécution au niveau dudit client est compatible avec ledit objet demandé ; et
une fonction de construction pour construire un environnement d'exécution au niveau dudit client qui est compatible avec ledit objet demandé ;

dans lequel des fonctions supplémentaires sont téléchargées à partir dudit serveur au niveau dudit client si nécessaire.

35. Dispositif de traitement de données de client selon l'une quelconque des revendications 25 à 34, dans

lequel ledit client comprend :

un environnement d'exécution ayant des objets de pilote de dispositif ;
un environnement d'exécution ayant des objets de système ; et
un environnement d'exécution ayant des objets d'environnement d'exécution ;

dans lequel lesdits objets de pilote de dispositif, lesdits objets de système et lesdits objets d'environnement d'exécution qui sont fournis au niveau dudit client sont les fonctions nécessaires pour créer l'environnement correspondant et où des objets de pilote de dispositif ou des objets de système plus hautement fonctionnels sont téléchargés à partir dudit serveur au niveau dudit client si nécessaire.

36. Dispositif de traitement de données de client selon l'une quelconque des revendications 25 à 35, dans lequel ledit serveur passe une structure de caractéristique audit client qui pointe vers une description dudit objet demandé et en outre pointe vers une description dudit environnement d'exécution compatible, et où ledit client utilise ladite structure de caractéristique pour déterminer un résultat passé audit serveur par ledit moyen de notification.

37. Système de communication comprenant un dispositif de serveur selon l'une quelconque des revendications 13 à 24 et un dispositif de traitement de données de client selon l'une quelconque des revendications 25 à 36.

38. Logiciel d'ordinateur ayant un code de programme pour réaliser un procédé selon l'une quelconque des revendications 1 à 13.

39. Support de fourniture de données par lequel le logiciel d'ordinateur selon la revendication 38 est fourni.

40. Support selon la revendication 39, le support étant un support de transmission.

41. Support selon la revendication 39, le support étant un support de stockage.

FIG. 1

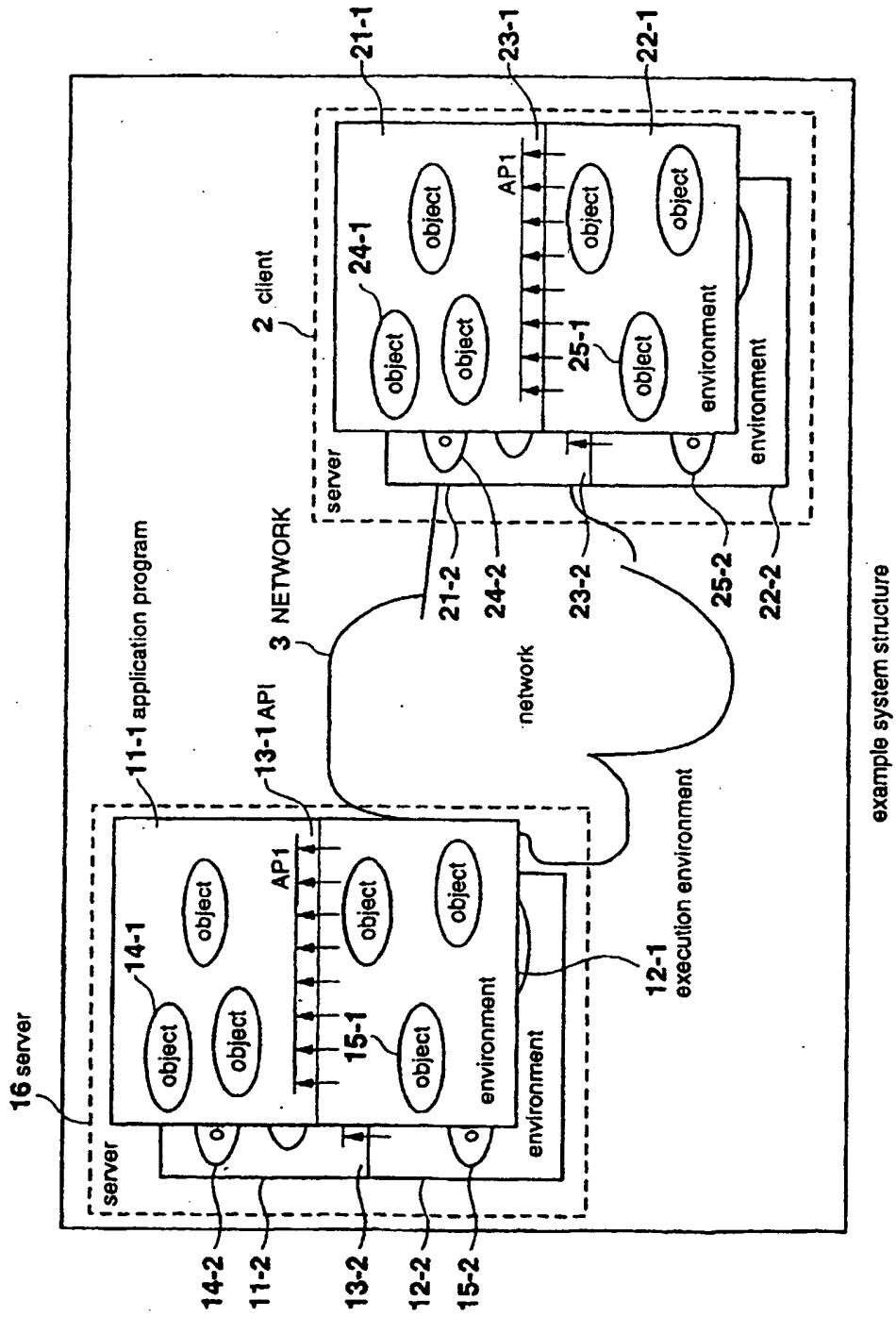
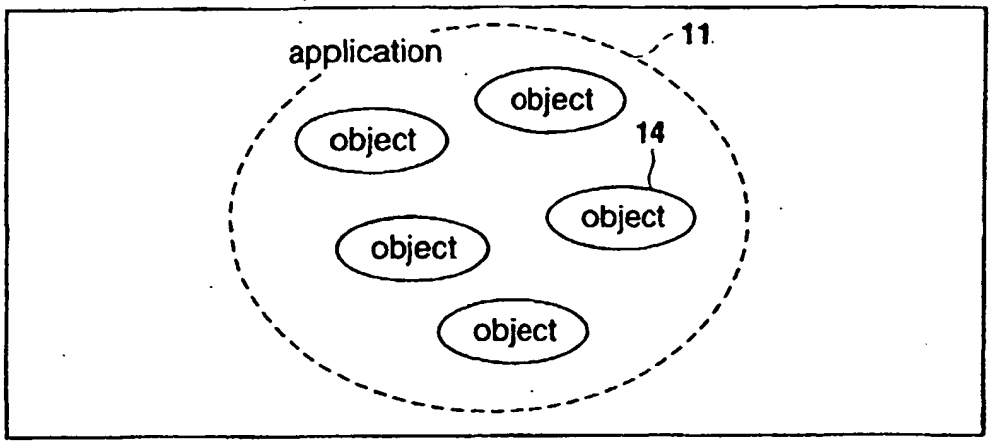
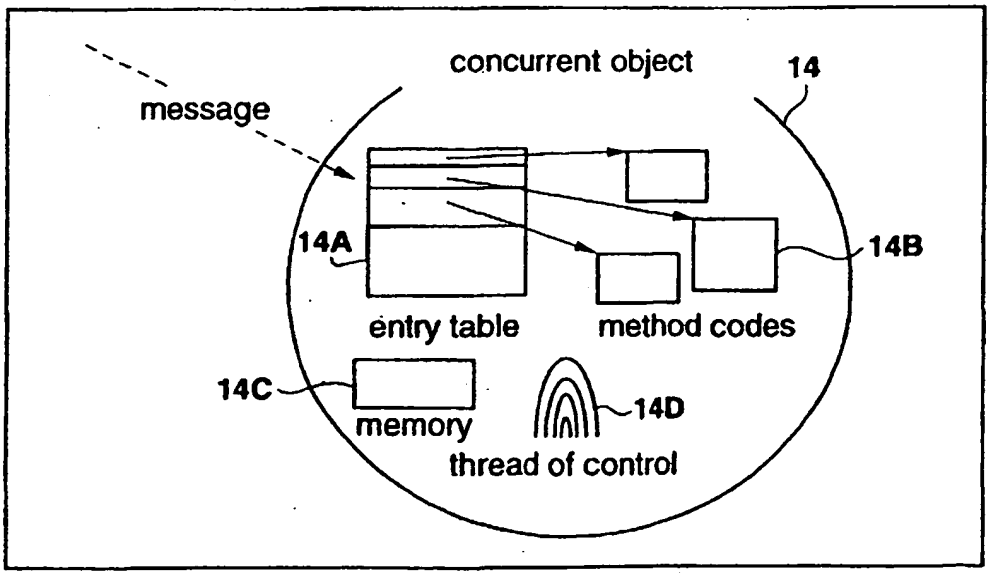

16 server
server
11-1 application program
13-1 API
AP1
14-1
object
object
object
object
object
15-1
environment
environment
12-1
execution environment
3 NETWORK
network
2 client
server
21-1
23-1
22-1
24-1
object
object
object
object
object
25-1
environment
environment
21-2
24-2
23-2
25-2
22-2
14-2
11-2
13-2
12-2
15-2
example system structure

FIG. 2



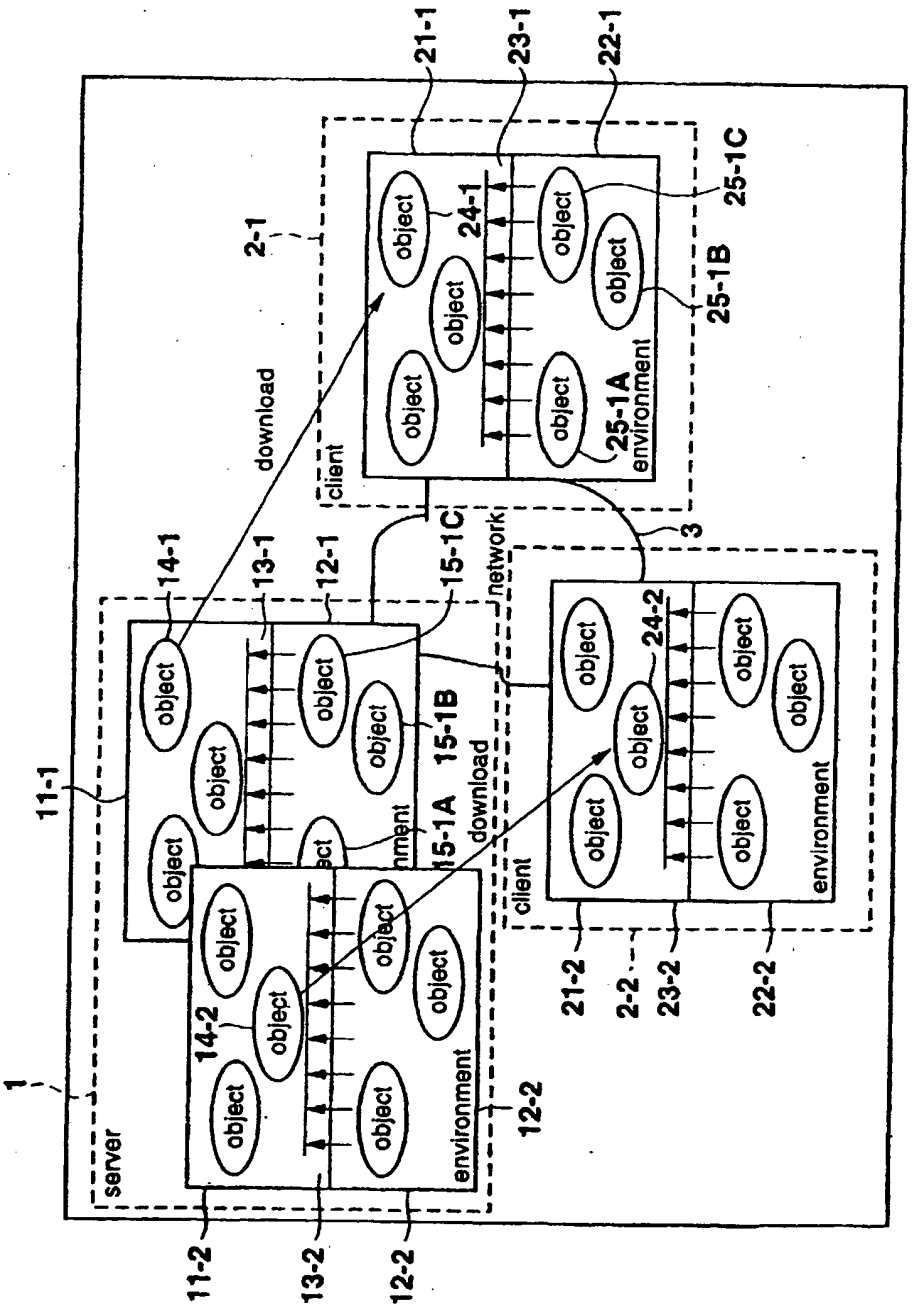
application program configuration

FIG. 3



concurrent object structure

FIG. 4



method of downloading objects from the server to clients of a plurality of vendors

FIG. 5

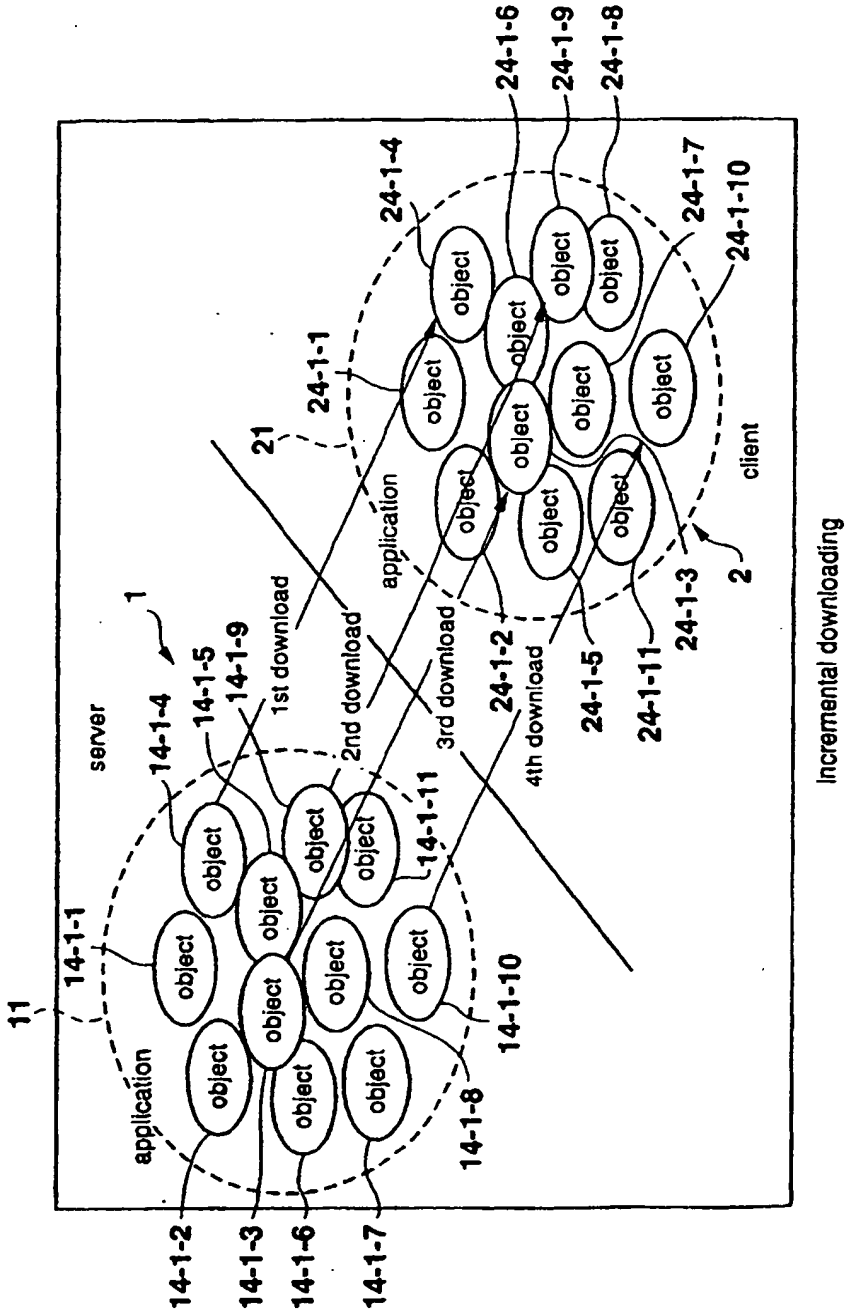

server
1
11
14-1-1
application
object
14-1-2
14-1-3
14-1-6
14-1-7
14-1-8
14-1-10
14-1-11
14-1-4
14-1-5
14-1-9
1st download
2nd download
3rd download
4th download
21
24-1-1
24-1-2
24-1-3
24-1-4
24-1-5
24-1-6
24-1-7
24-1-8
24-1-9
24-1-10
24-1-11
application
client
2
Incremental downloading

FIG. 6

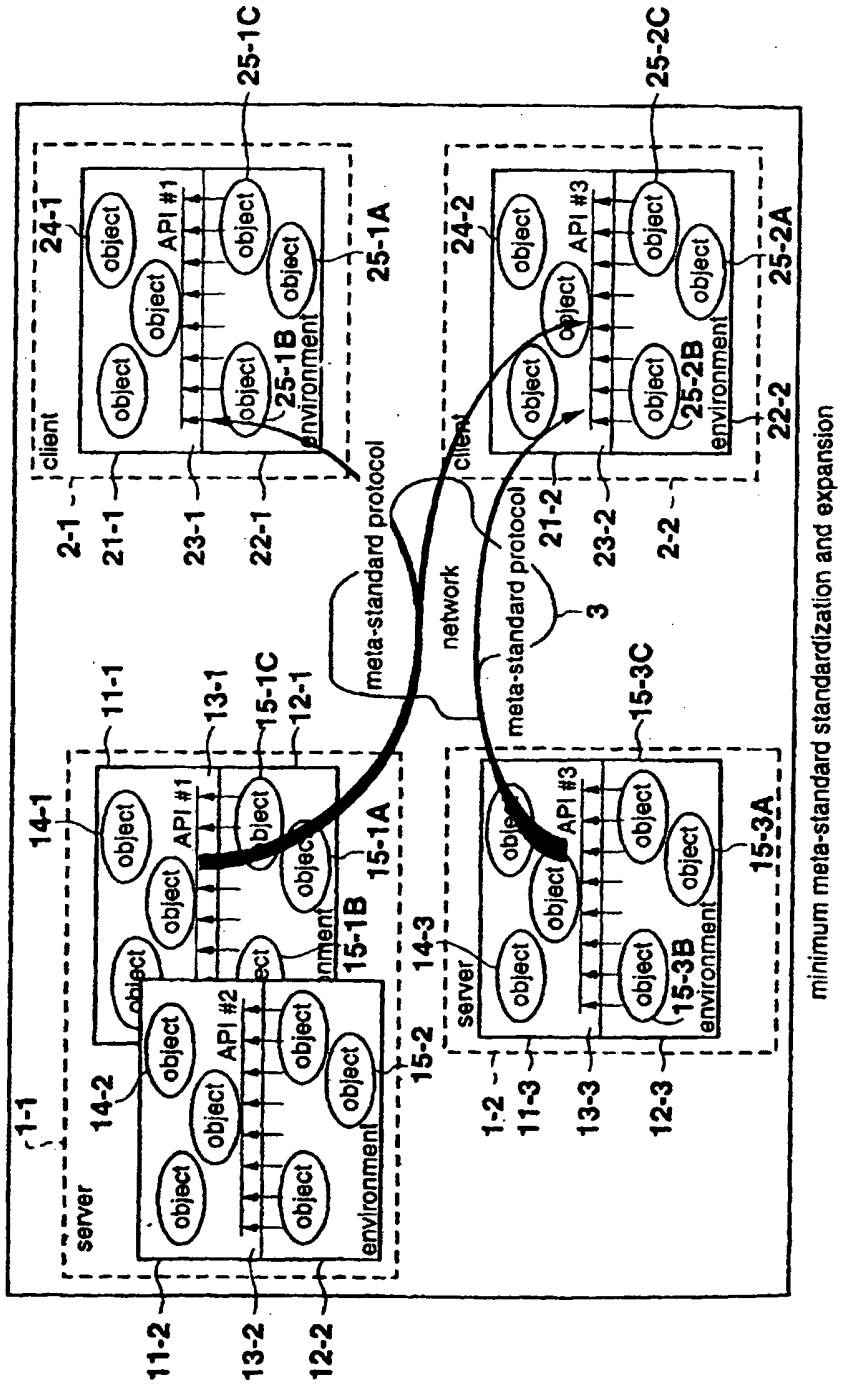

minimum meta-standard standardization and expansion
network
meta-standard protocol
meta-standard protocol
client
server
client
server
API #1
API #2
API #3
object
environment
25-1C
25-2C
24-1
24-2
25-1A
25-2A
25-1B
25-2B
25-3A
25-3B
25-3C
22-1
22-2
23-1
23-2
21-1
21-2
2-1
2-2
3
11-1
11-2
11-3
13-1
13-2
13-3
15-1C
12-1
12-2
12-3
15-1A
15-1B
15-2
15-3C
14-1
14-2
14-3
1-1
1-2

FIG. 7

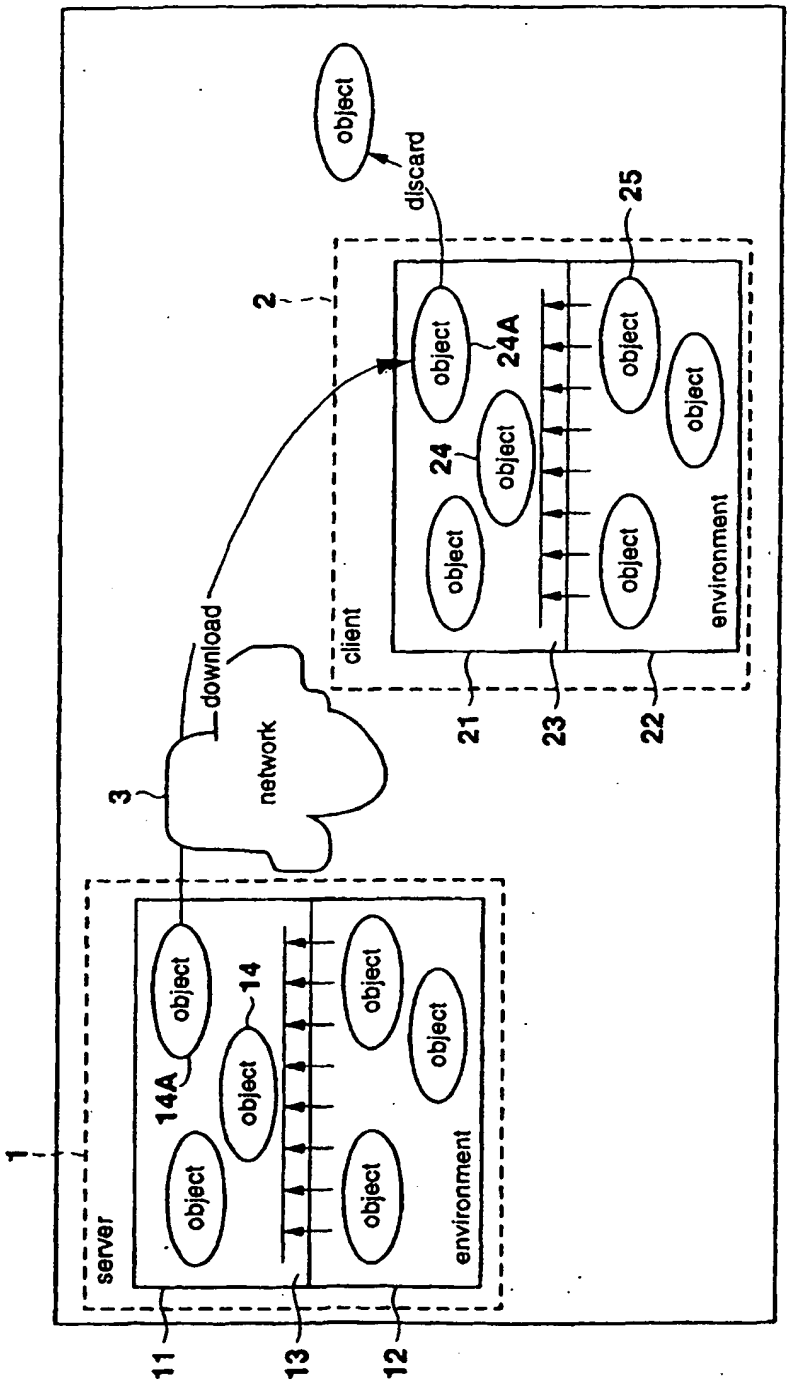

server
environment
object
14A
14
client
environment
24A
24
25
discard
download
network
3
1
2
11
13
12
21
23
22
dynamic changing of object

FIG. 8

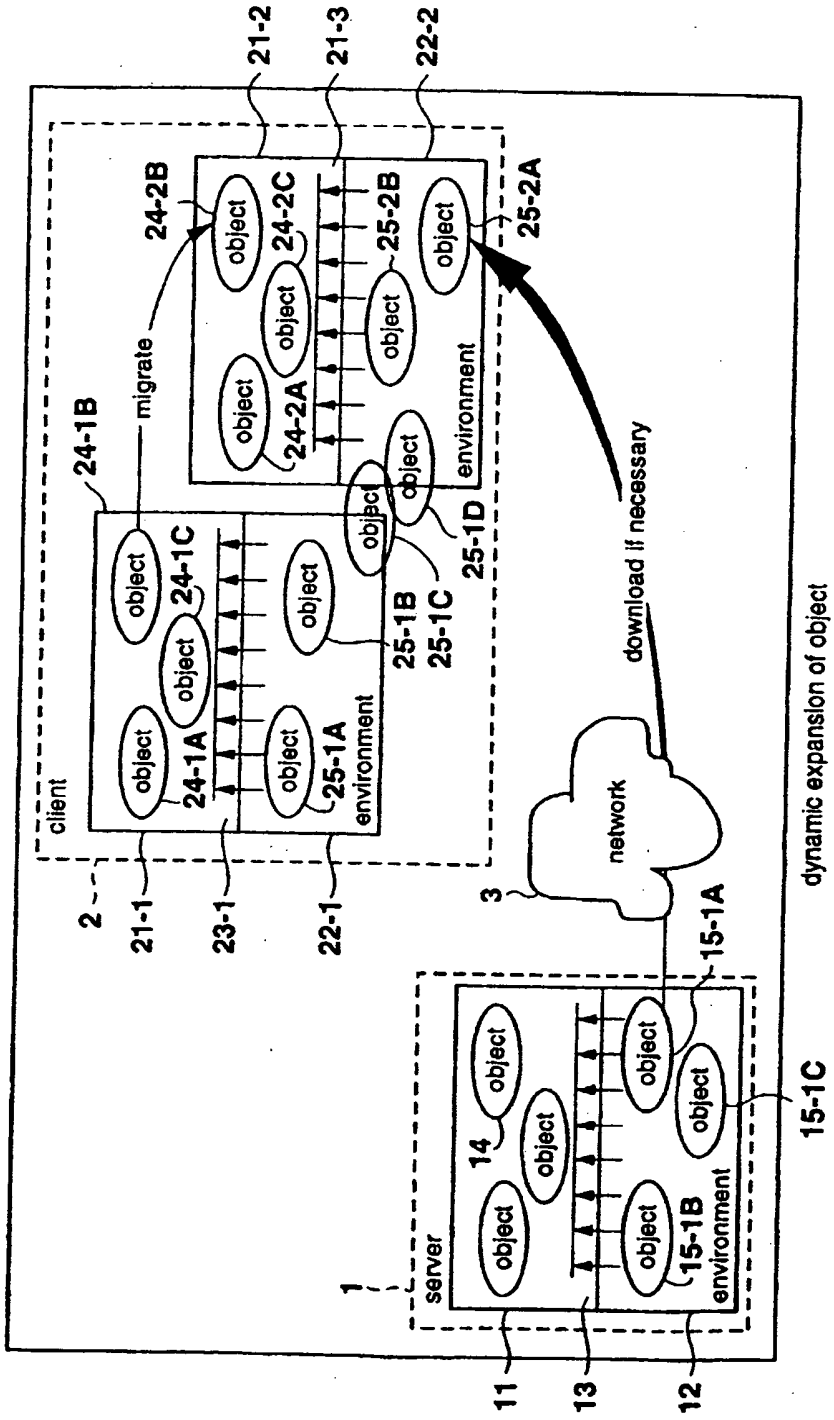

21-2
21-3
22-2
24-2B
24-2C
25-2B
25-2A
object
24-2A
environment
migrate
24-1B
24-1C
24-1A
25-1A
25-1B
25-1C
25-1D
client
download if necessary
network
dynamic expansion of object
2
21-1
23-1
22-1
3
15-1A
15-1C
server
14
15-1B
1
11
13
12

FIG. 9

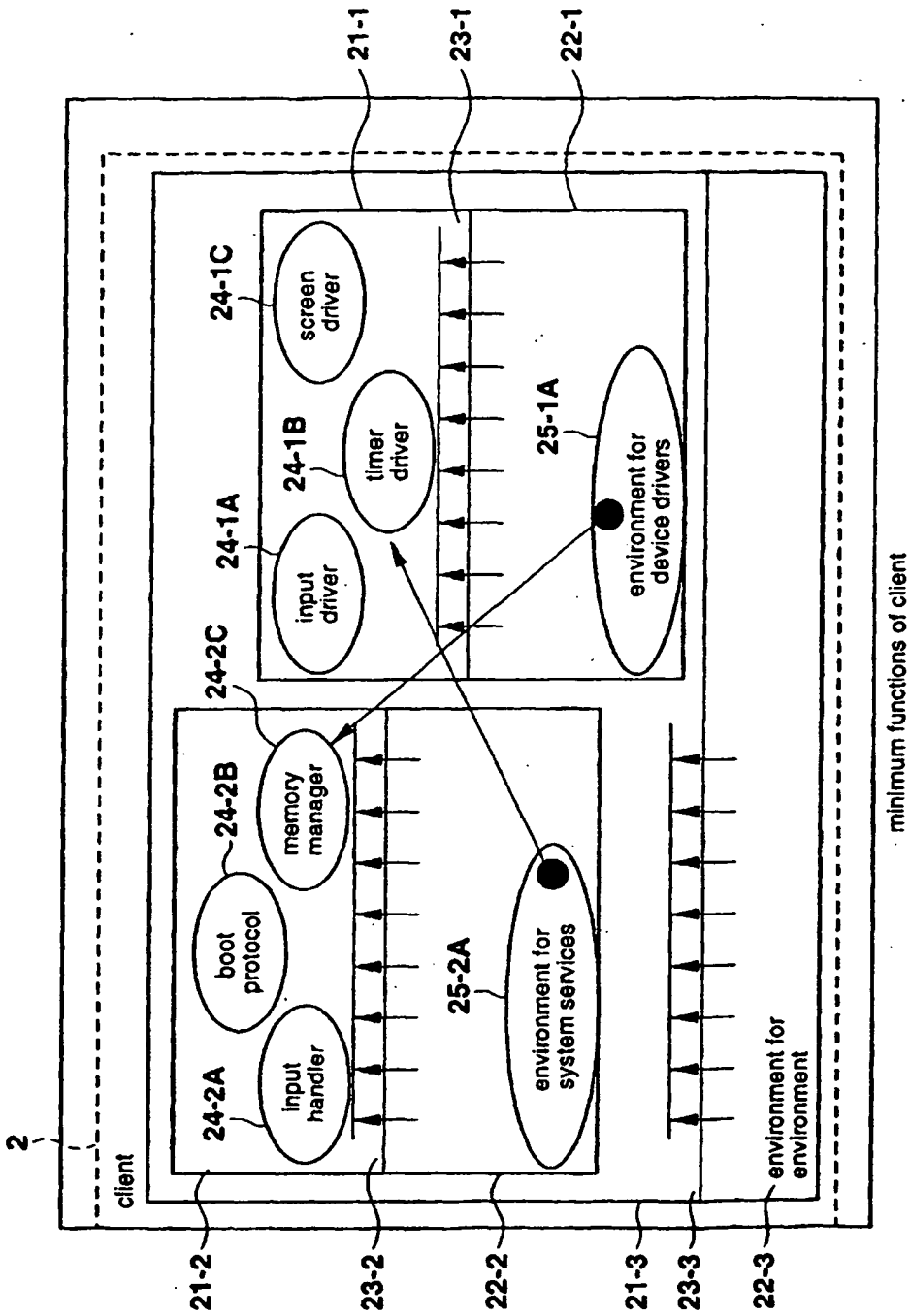
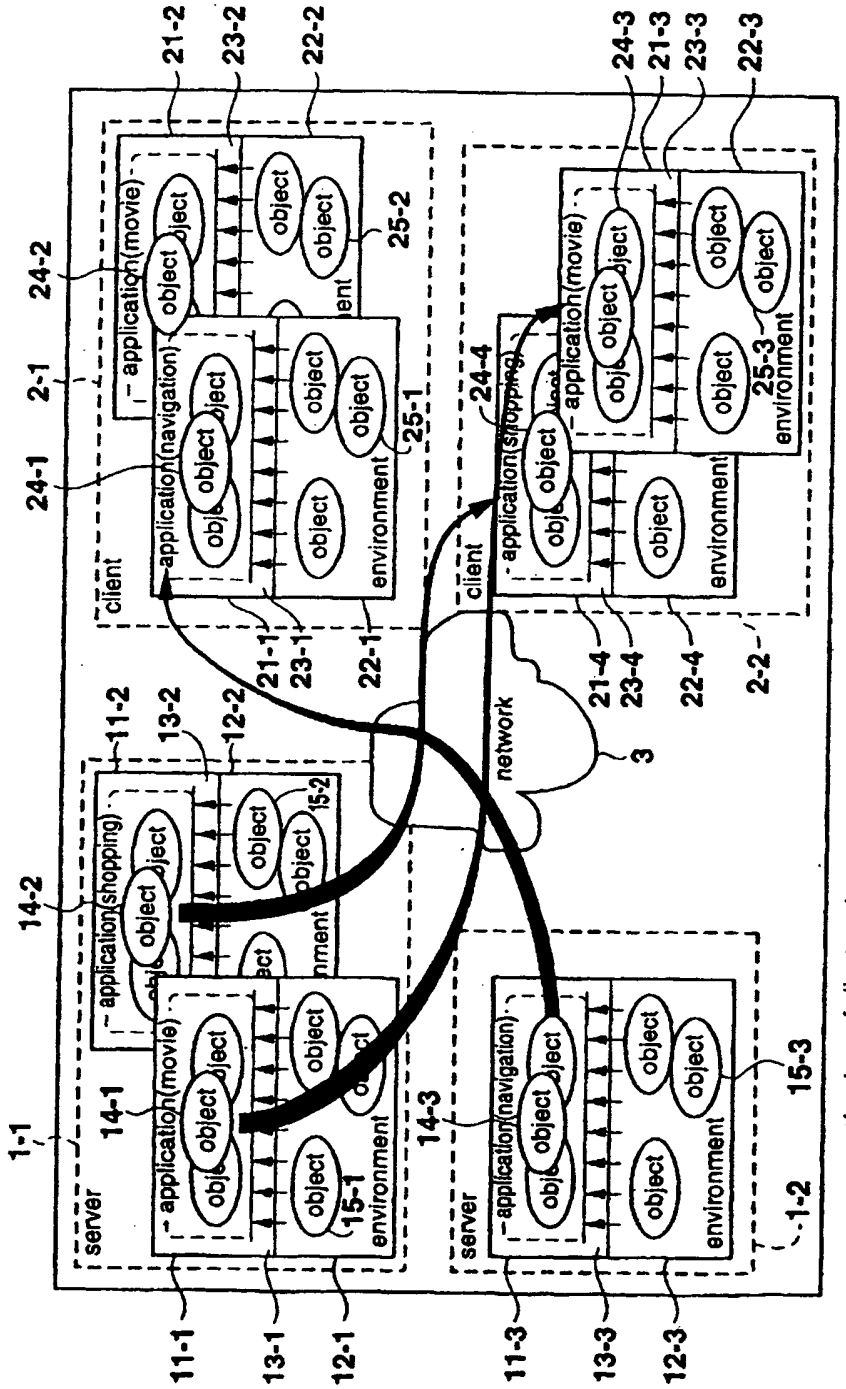

2
client
21-1
23-1
22-1
24-1C
screen driver
24-1B
timer driver
24-1A
input driver
25-1A
environment for device drivers
24-2C
memory manager
24-2B
boot protocol
24-2A
input handler
25-2A
environment for system services
environment for environment
minimum functions of client
21-2
23-2
22-2
21-3
23-3
22-3

FIG. 10



structure of client environment suitable for applications and dynamic restructuring.

FIG. 11

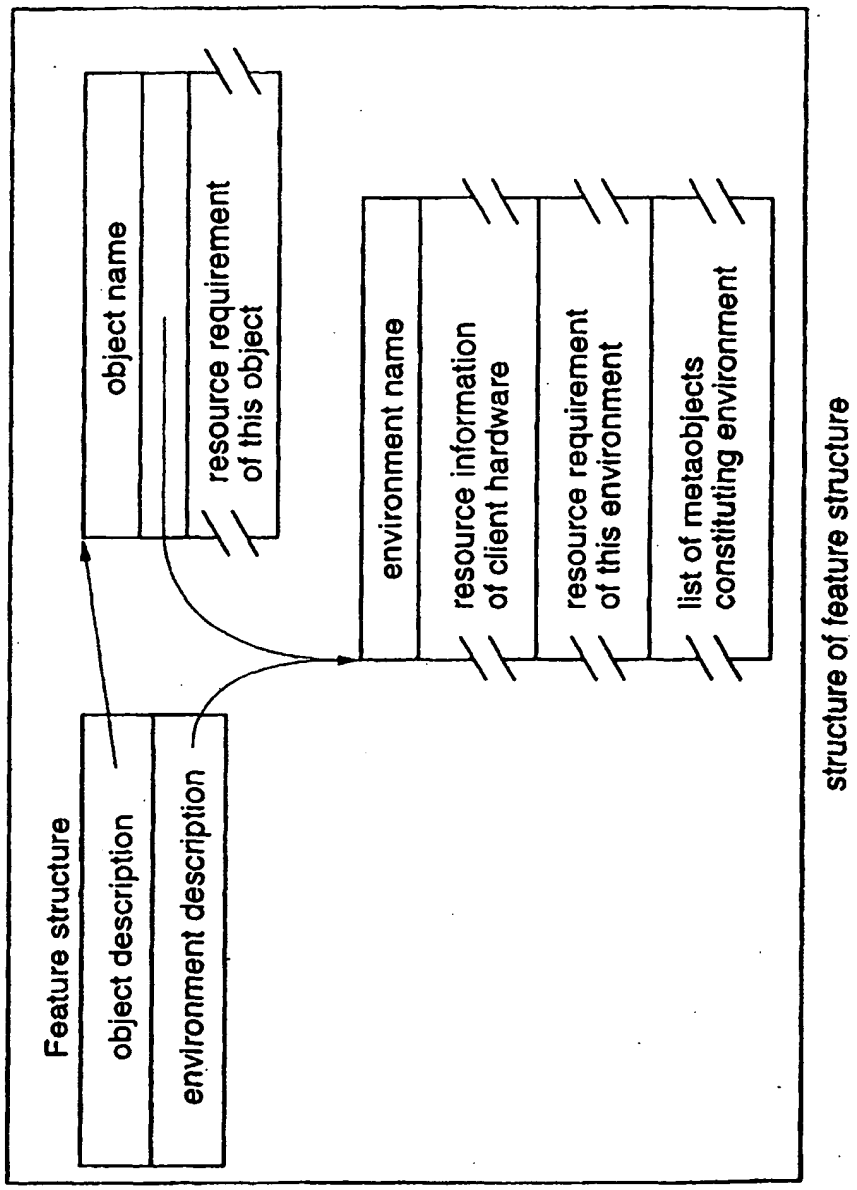
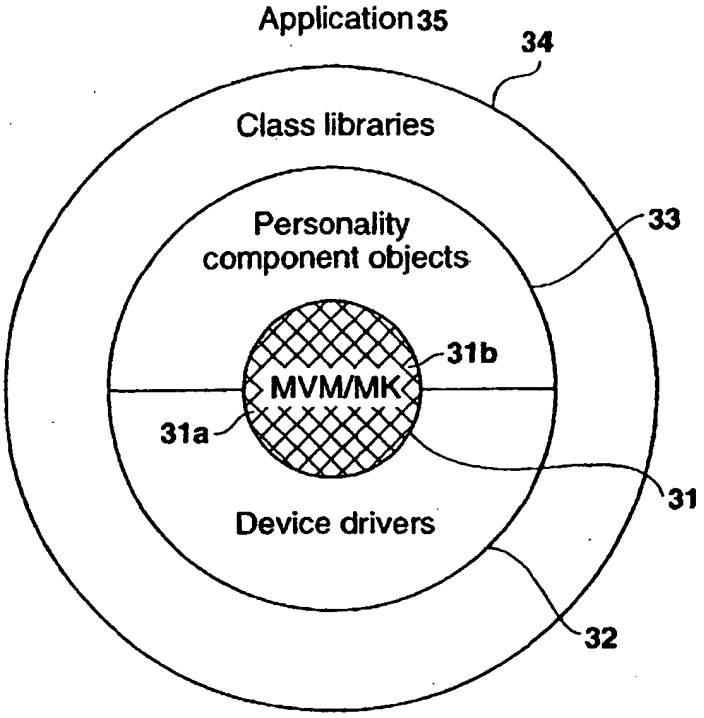

Feature structure
object description
environment description
object name
resource requirement of this object
environment name
resource information of client hardware
resource requirement of this environment
list of metaobjects constituting environment
structure of feature structure

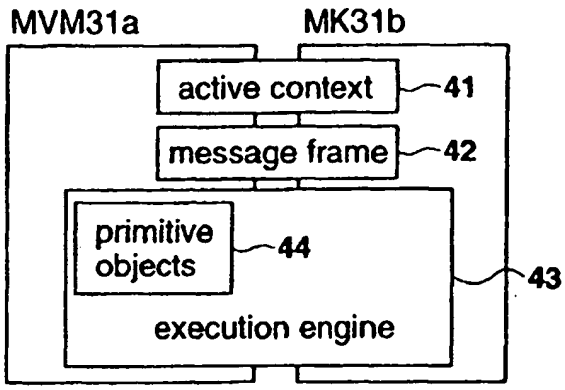
FIG. 12



MK : Micro Kernel
MVM : Micro Virtual Machine

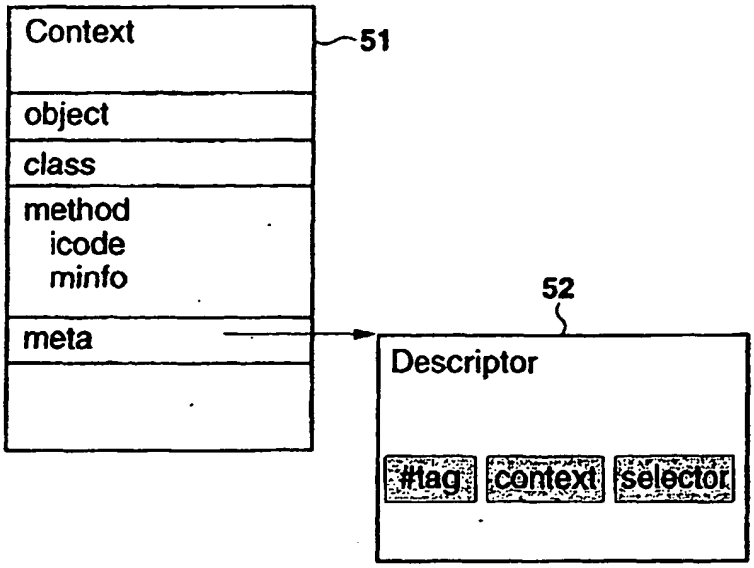
structure of system functions

FIG. 13



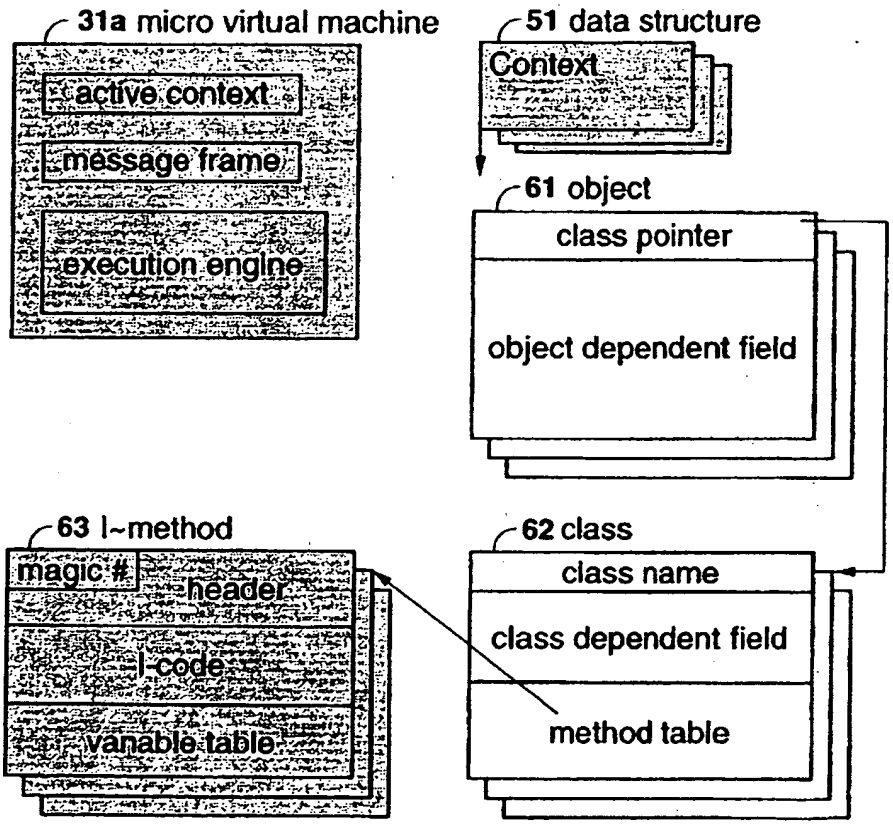
logical structure of MVM and MK

FIG. 14



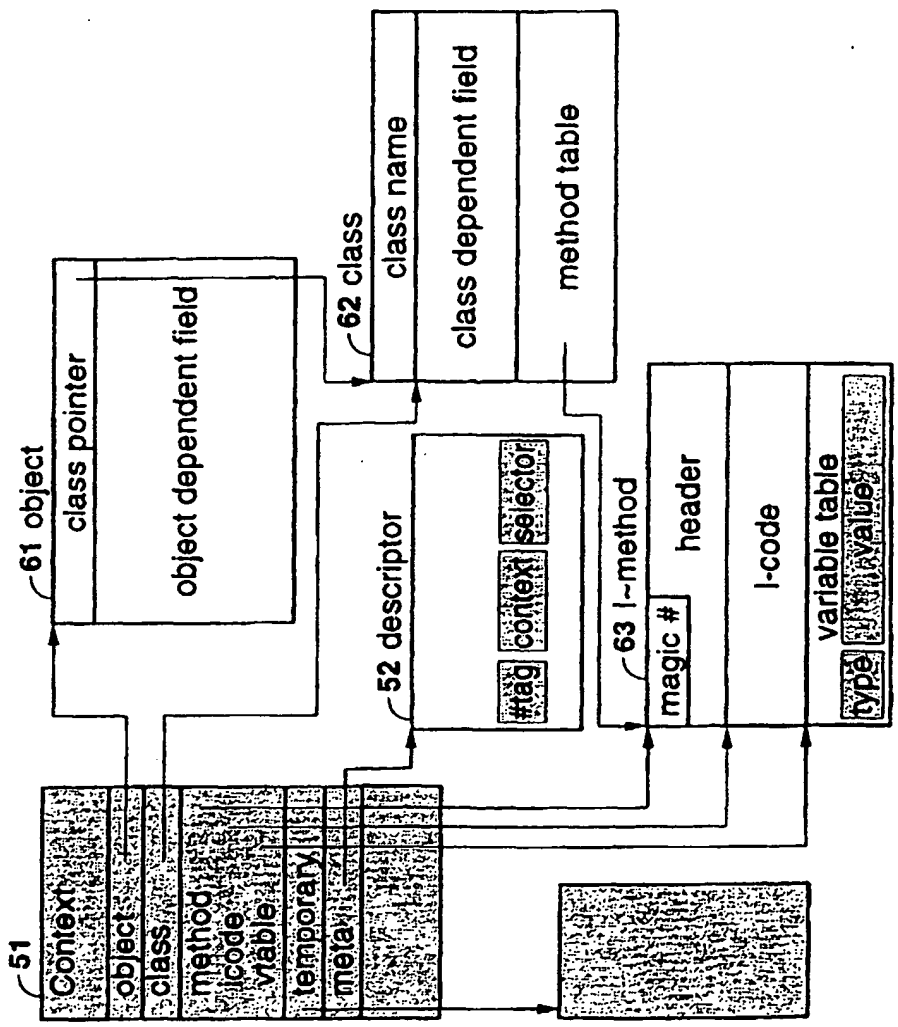
logical structure of Context and Descriptor

FIG. 15



overall structure of MVM

FIG. 16



data structure of Context and surroundings

FIG. 17

type	description
T_INVALID	invalid entry
T_PRIMITIVE	The value field denotes a primitive object, which is a dependent object on MVM
T_POINTTER	The value field contains a pointer to an object

type of variable table entry

FIG. 18

class	P_CLASS	P_BODY	description
Integer	P_INTEGER	immediate	integer number
Float	P_FLOAT	immediate	IEEE 754 32bits floating point number
Double Float	P_DFLOAT	immediate	IEEE 754 64bits floating point number
Boolean	P_BOOLEAN	immediate	TRUE or FALSE
String	P_STRING	address to heap	sequence of UNICODE characters
Array	P_ARRAY	address to heap	sequence of data of the same type
Context	P_CONTEXT	CName	representation of object execution
Jump	P_JUMP	I-code index	target address to which execution resumes
Pointer	P_POINTER	address to memory	adress
Assign	P_ASSIGN	n/a	duplication of a variable entry
Mailer	P_MAILER	n/a	invocation of a method
Undef	P_UNDEF	n/a	undefined primtive object

primitive object

EP 0 753 811 B1

FIG. 19

19-1

```
struct Message {
  long length ;
  any body [ ];
};
exception outrange {long position} ;
exception overflow {} ;
exception undef {int primitive; int selector} ;
exception zerodiv {} ;

Interface Integer {
  Integer operator+(in Integer value)
    raise(overflow);
  Integer operator-(in Integer value)
    raise(overflow);
  Integer operator*(in Integer value);
  Integer operator/(in Integer value)
    raise(zerodiv);
  Integer remainder(in Integer value);
  Integer and(in Integer value);
  Integer or(in Integer value);
  Integer xor(in Integer value);
  Integer not(void);
  Integer operator=(in Float value);
  Integer operator=(in DoubleFloat value);
  Boolean opetator<(in Integer value);
  Boolean opetator<=(in Integer value);
  Boolean opetator==(in Integer value);
};
interface Float {
  Float operator+(in Float value)
    raise (overflow, underflow);
  Float operator-(in Float value)
    raise (overflow, underflow);
  Float operator*(in Float value)
    raise (overflow, underflow);
  Float operator/(in Float value)
    raise (overflow, underflow, zerodiv);
  Float operator=(in Integer value);
  Float operator=(in DoubleFloat value);
    raise (overflow);
  Boolean opetator<(in Float value);
  Boolean opetator<=(in Float value);
  Boolean opetator==(in Float value);
```

primitive object interface

FIG. 20

19-2

```
interface DoubleFloat{
        DoubleFloat operator+(in DoubleFloat value)
              raise(overflow, underflow);
        DoubleFloat operator-(in DoubleFloat value)
              raise(overflow, underflow);
        DoubleFloat operator*(in DoubleFloat value)
              raise(overflow, underflow);
        DoubleFloat operator/(in DoubleFloat value)
              raise(overflow, underflow, zerodiv);
        DoubleFloat operator=(in Float value)
              raise( );
        Boolean operator<(in DoubleFloat value)
              raise( );
        Boolean operator<=(in DoubleFloat value)
              raise( );
        Boolean operator==(in DoubleFloat value)
              raise( );
};
interface Boolean {
        Boolean not (void)
};
interface String {
        String operator+(in String string);
        String substring(in Integer position, in Integer length);
              raise(outrange);
        Integer length(void);
};
interface Array {
        void put (in Integer Index, in any value);
              raise(outrange);
        any get (in Integer Index)
              raise(outrange);
        Integer length (void);
};
interface Context {
  ...accessers...
};
interface Jump {
        void eval (void)
              raise(outrange);
        void evalT (Boolean cond)
              raise(outrange);
        void evalF (Boolean condition)
              raise(outrange);
};
```

primitive object interface (continued)

FIG. 21

```
19-3
interface Pointer {
        void put (in Address pos, in byte value)
                exception outrange;
        byte get (in Address pos)
                exception outrange;
        void put (in Address pos, in word value)
                exception outrange;
        word get (in Address pos)
                exception outrange;
        void put (in Address pos, in long value)
                exception outrange;
        long get (in Address pos)
                exception outrange;
        void put (in Address pos, in longlong value)
                exception outrange;
        longlong get (in Address pos)
                exception outrange;
};
interface Assing{
        void operator=(in int position1, in int position2);
interface Mailer{
        void call(in ID target, Integer select, in Message msg);
        void reply( );
};
interface undef
};
```

primitive object interface (continued)

FIG. 22

name	description
OP_M	execute the operation specified by the first argument. This operation is processed either by personality or by a primitive object. (operands:operation, number of operands,...)
OP_R	give control back to the object when the operation requested by OP_M returns (operands:target, return arguments)

I-code instruction set

FIG. 23

```
interface MetaCore{
    mcError M (in long method, in Message msg);
    mcError R (in CName ctxt, in long method, in Message msg);
```

Micro Kernel Interface

THIS PAGE BLANK (USPTO)